LaTeX für Studenten

Kurze Einführung in die Textformatierung mit dem LaTeX-Satzsystem

Jürgen Plate 31. Mai 2015

Inhaltsverzeichnis

1

Textbearbeitung mit LaTeX

LaTeX ist ein System zum Setzen (Layouten) von Texten. Sie können damit vom Brief bis hin zu einem Buch beinahe jeden beliebigen Text gestalten. LaTeX ist wegen seines herausragenden Formelsatzes vor allem in der (natur-)wissenschaftlichen Welt sehr beliebt. Zahllose Diplomarbeiten, Dissertationen und wissenschaftliche Veröffentlichungen wurden und werden damit verfasst – so auch dieser Text.

Dieses Skript bietet einen Schnelleinstieg in LaTeX und fasst die wichtigsten LaTeX-Anweisungen zusammen. Außerdem finden Sie einige Hintergrundinformationen darüber, wie LaTeX Schriften nutzt, wie Sie aus LaTeX-Dokumenten PostScript- und PDF-Dateien erzeugen etc. Der Platz gestattet es leider nicht, auf die vielen vorzüglichen Makropakete einzugehen, die es für nahezu jeden Zweck gibt – hier müssen wir auf die weiterführende Literatur und den Server von Dante e. V. (www.dante.de) verweisen.

1.1 Einführung

Die besonderen Vorzüge von LaTeX gegenüber anderen Programmen bestehen im grandiosen Formelsatz ($\pi \sum_{i=1}^{n} \frac{a_i x^i}{i!}$) und in der überragenden Satzqualität (automatischer Zeichenausgleich). Beispielsweise wird in „Vektor“ das „e“ näher an das „V“ gerückt. LaTeX hat aber auch Nachteile: Die Bedienung des Programms ist steinzeitlich. Scheinbar triviale Dinge wie manche Trennungen oder Seitenumbrüche müssen oft manuell verändert werden, wenn LaTeX standardmäßig nicht die gewünschten Resultate liefert.

Kurz einige Worte zur Herkunft von TeX und LaTeX: Genau genommen ist LaTeX ein Makropaket, das das Satzprogramm TeX erweitert. TeX ist aber noch komplizierter anzuwenden als LaTeX und steht deswegen im Schatten seiner Erweiterung LaTeX.

TeX wurde in seiner ursprünglichen Form von Donald Knuth programmiert, das dazugehörige Makropaket LaTeX von Leslie Lamport. Seit Leslie Lamport die Weiterentwicklung von LaTeX mit Version 2.09 eingestellt hat, sind es vor allem Frank Mittelbach und Rainer Schöpf, denen die aktuelle Version LaTeX 2_ε und die Pläne zur Weiterentwicklung in Richtung LaTeX 3 zu verdanken sind.

LaTeX von Leslie Lamport ist zwar im Vergleich zu TeX relativ einfach zu erlernen, es kann aber nicht mit dem Komfort gewöhnlicher Textverarbeitungsprogramme wie OpenOffice Writer, KWord oder Microsoft Word mithalten – was das WYSIWYG (*What You See Is What You Get*) betrifft, denn bei LaTeX sieht man zunächst nur Text mit Befehlen zur späteren Struktur und Formatierung. Das Generieren des Layouts erfolgt in einem gesonderten Schritt (beinahe so wie bei der Erstellung eines Programm-Quelltextes und der späteren Kompilierung). Dafür können die Hände beim Schreiben auf der Tastatur bleiben. Bei LaTeX handelt es sich also eher um ein Programm vom Typ WYSIWYM (*What You See Is What You Mean*) – und das ist sehr viel besser. Es steht so die Konzentration auf den Inhalt statt auf die Form im Vordergrund, denn Änderungen am Layout sind nicht ganz trivial. [1]

[1] Das Gefrickel am Text fällt damit auch weg, was unheimlich Zeit spart.

Der größte Vorteil von LaTeX liegt jedoch darin, dass der Dokumenten-Quelltext in reinem ASCII erzeugt wird und man so auch Dokumente noch verwerten kann, die älter sind. (Versuchen Sie mal ein zehn Jahre altes Word-Dokument zu bearbeiten.) Außerdem ist LaTeX plattformunabhängig – man kann die Dokumente problemlos zwischen Linux, Windows, Mac OS oder anderen Betriebssystem- und Rechnerplattformen austauschen. Aber wir geben auch zu, dass die Arbeit mit LaTeX nach den ersten leichten Schritten fummelig werden kann, etwa wenn es um Tabellen oder ein bestimmtes Layout geht. Dafür werden auch Bücher mit 1200 Seiten und zahllosen Abbildungen ohne Murren verarbeitet. Auch ist eine gute Silbentrennung Bestandteil des Systems.[2]

Für die Super-Profis gibt es noch einen weiteren Vorteil. Man kann LaTeX-Code auch per Programm erzeugen und so beispielsweise die Ergebnisse einer Datenbankabfrage in LaTeX-Code einfügen und daraus ein super tolles PDF-Dokument erzeugen – und das ohne jeden manuellen Eingriff.

Die Vorteile von LaTeX:

- Viele Formatvorlagen liegen bereits in LaTeX vor
- ausgezeichneter mathematischer Formelsatz, der alle Möglichkeiten bietet
- portables, offenes Dateiformat (Windows, Linux, Mac)
- Unterstützung großer Dokumente
- kompakte Eingabedateien
- mit PdfLaTeX direkte Ausgabe von PDF-Dokumenten
- Open Source – frei verfügbar
- Formatierung unabhängig von Ausgabegerät
- viele gute Bücher und Dokumentationen
- ganz viele Automatismen (Makros, Umgebungen, siehe später)
- vorzügliche Silbentrennung

Die Nachteile von LaTeX:

- Bedienung gewöhnungsbedürftig (kein WYSIWYG, ggf. Makefile oder Batch-Datei)
- Editier-Compilier-Zyklus
- Änderungen am Layout nicht ganz trivial (was man auch als Vorteil sehen kann)

Weiterführende Dokumentation: Dieses Kapitel kann nur eine LaTeX-Einführung geben. Darüber hinausgehende Informationen finden Sie in den zahlreichen LaTeX-Büchern. Besonders zu empfehlen sind dabei die Bücher von Helmut Kopka und der *LaTeX-Begleiter* von Michael Goosens, Frank Mittelbach und Alexander Samarin.

Zusammen mit LaTeX sind eine Menge Dokumentationsdateien installiert, bei aktuellen Versionen überwiegend im PDF-Format. Entsprechende Dateien finden Sie mit den folgenden Kommandos:

```
user$ find /usr/share/texmf -name '*.pdf'
user$ locate '/usr/share/texmf/*.pdf'
```

Sehr hilfreich sind auch die folgenden Seiten im Internet:

http://www.dante.de/faq/de-tex-faq/
http://www.ctan.org/tex-archive/info/german/
http://www.ctan.org/tex-archive/info/german/LaTeX2e-Kurzbeschreibung/l2kurz2.pdf

[2] Bei anderen Textverarbeitungen gibt es keine Silbentrennung oder es wird oft „vergessen“ sie zu installieren – entsprechend löchrig sieht der Blocksatz dann aus.

1.2 LaTeX-Distributionen

1.2.1 teTeX

Das Programm LaTeX kann nicht isoliert gesehen werden, sondern muss in seiner Gesamtheit mit diversen Style-Dateien, dem Zusatzprogramm Metafont, dessen Schriftarten etc. betrachtet werden. Diese Gesamtheit wird als „LaTeX-Distribution“ bezeichnet.

Die zurzeit am weitesten verbreitete LaTeX-Distribution für Linux ist das von Thomas Esser gewartete teTeX (*http://www.tug.org/teTeX/*). Alle Aussagen in diesem Kapitel beziehen sich auf die teTeX-Version 2.0. Da nur wenige Anwender wirklich alle TeX- und LaTeX-Erweiterungen benötigen, wird teTeX bei den meisten Linux-Distributionen in mehrere Pakete zerlegt, von denen per Default meist nur wenige installiert werden. Dabei werden in der Regel folgende Verzeichnisse genutzt:

`/etc/texmf/`	Konfigurationsdateien
`/usr/share/texmf/`	TeX- und LaTeX-Dateien
`/var/lib/texmf`	veränderliche Dateien
`/var/cache/fonts/`	dynamisch erzeugte Font-Bitmaps (*.pk)

In der teTeX-Distribution sind zwar schon sehr viele Pakete und Werkzeuge vereint, aber es gibt noch weit mehr LaTeX-Utilities, -Erweiterungen und -Speziallayouts (etwa zum Dokumentieren von Schachstellungen, zum Notensatz etc.). Diese Erweiterungen werden vom CTAN (*Comprehensive TeX Archive Network*) gesammelt:

http://www.latex-project.org/
http://www.dante.de
http://dante.ctan.org/CTAN

1.2.2 proTeXt

Dies ist eine einfach zu installierende TeX-Distribution für Windows, die auf MiKTeX basiert. Nach dem Download führt die Installation über ein kurzes PDF-Dokument (wahlweise in Englisch, Französisch, Deutsch und Italienisch) mit anklickbaren Links durch die Installation der verschiedenen Komponenten. Mit dabei sind einige Zusatzprogramme u. a. TeXstudio für die Bearbeitung des Quelltextes, Ghostscript zur Erzeugung von Postscript- und PDF-Dokumenten und Ghostview als PS-Betrachter. proTeXt ist Teil der TeX-Collection.

1.2.3 TeX Collection

Jedes Jahr im Sommer gibt der Dante e. V. eine DVD mit dem Titel „TeX Collection“ heraus. Der Ladenpreis beträgt ca. 10 Euro, zu Bestellen ist sie bei Lehmanns Fachbuchhandlung. Die TeX Collection bietet auf einer einzelnen DVD vier Produkte für TeX-Nutzer:

- **proTeXt** ist ein TeX-System für Microsoft Windows (siehe oben). Vorrangiges Ziel ist eine einfache Installation auch für weniger erfahrene Nutzer. Das System basiert auf MiKTeX von Christian Schenk, ergänzt um einige Zusatzprogramme.
- **TeX Live** ist ein ready-to-run-TeX-System für Linux, Unix und Mac OS X. Es basiert auf Web2C und teTeX mit Ergänzungen. Sie können es wahlweise direkt von der DVD laufen lassen oder auf Ihrer Festplatte installieren. Derzeit enthält TeX Live Binaries für folgende Systeme:

```
GNU/Linux (i386, x86\_64, PowerPC, Alpha, Sparc)
MacOSX (PowerPC, i386)
FreeBSD (i386, x86\_64)
OpenBSD (i386)
HP-UX
SGI IRIX
IBM AIX (PowerPC)
Sun Solaris (Sparc)
MS Windows (32bit)
```

- **MacTeX** ist ein einfach zu installierendes TeX-System für Mac OS X, das neben TeX Live zusätzliche Programme wie TeXShop enthält.

- **CTAN** CTAN steht für „Comprehensive TeX Archive Network“ und ist ein weltweiter Verbund von Internet-Servern zur Verbreitung von TeX-Software. Auf der DVD befindet sich ein kompletter Abzug vom deutschen CTAN-Knoten `dante.ctan.org`. Darin befinden sich so ziemlich alle aktuellen Makropakete.

1.3 LaTeX und die wichtigsten Hilfsprogramme

LaTeX ist ein Satzprogramm und kein Textverarbeitungsprogramm. Der Unterschied besteht darin, dass LaTeX nicht mit einem eigenen Editor ausgestattet ist. Vielmehr muss der zu setzende Text als gewöhnliche Textdatei mit einem beliebigen Editor geschrieben werden. Daraus ergibt sich auch, dass LaTeX kein WYSIWYG-Programm ist – ganz im Gegenteil: Sämtliche Satzanweisungen müssen in einer ziemlich unübersichtlichen Syntax im Text angegeben werden. Wenn Sie beispielsweise ein Wort kleiner schreiben möchten, lautet die LaTeX-Syntax hierfür `{\small klein}`. Bild 1.1 zeigt den prinzipiellen Ablauf bei Erstellen eines Dokuments.

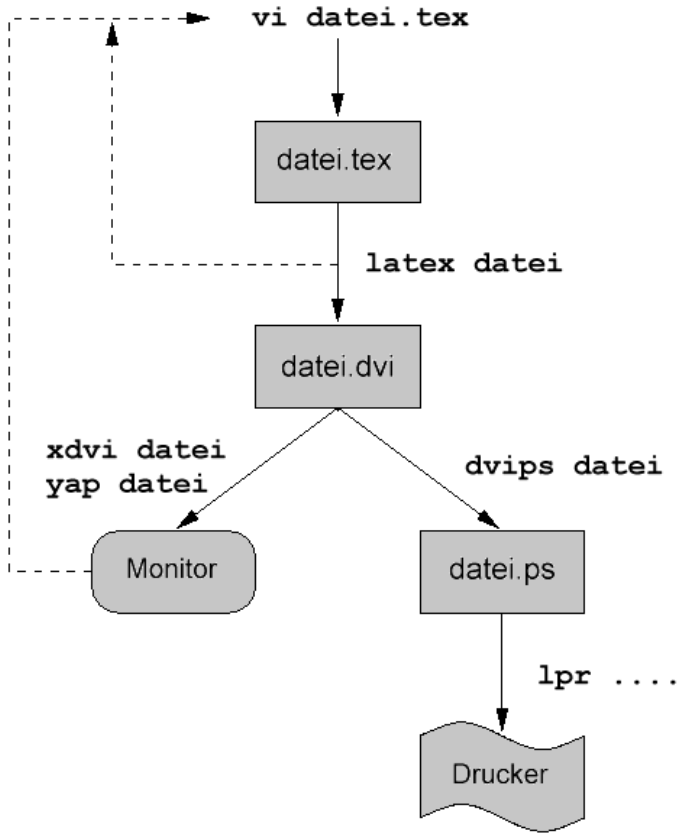

Bild 1.1: Prinzipieller Ablauf beim Erstellen eines LaTeX-Dokuments

1.3.1 Texterfassung

Zur Texteingabe können Sie grundsätzlich jeden beliebigen Editor verwenden. Besonders gut geeignet sind natürlich Editoren, die LaTeX verstehen, LaTeX-Schlüsselwörter farbig hervorheben und eventuell auch bei der Übersetzung der LaTeX-Datei und der Fehlersuche behilflich sind (z. B. vi, emacs, nano, kate). Noch mehr Komfort bieten LaTeX-Umgebungen wie das Programm `texmaker` (*http://www.xm1math.net/texmaker/download.html*).

Texmaker ist ein plattformunabhängiger Unicode-Editor für das Erstellen von LaTeX-Dokumenten. Die Software wird unter der GPL veröffentlicht. Der Editor richtet sich insbesondere an LaTeX-Anfänger, denen mit Hilfe von Assistenten die Erstellung von Dokumenten erleichtert werden soll.

Texmaker kann Dokumente kompilieren und anzeigen. Das Programm verfügt über Syntaxhervorhebung und einen Assistenten zur einfachen Erstellung von Tabellen. Sonderzeichen können per

Mausklick als LaTeX-Befehl eingefügt werden. Das Programm unterstützt Unicode und enthält eine integrierte Anzeige von DVI-, PostScript-, PDF- und HTML-Dateien. Das Programm verfügt über einen Sitzungsmanager, mit dem die Liste der aktuell geöffneten Dateien in einer Sitzungs-Datei gespeichert werden und später wieder geladen werden können. Eine sogenannte Masterdatei eines Projekts wird immer unabhängig von der gerade geöffneten übersetzt.

TeXstudio (ehemals TeXmakerX) ist ein weiterer plattformunabhängiger Editor für die Erstellung von LaTeX-Dokumenten. Es ist einer von mehreren Forks des LaTeX-Editors Texmaker. Im Unterschied zu Texmaker bietet TeXstudio eine größere Interaktivität mit dem Quelltext, zum Beispiel Aktualisierung der Strukturansicht während des Tippens, Interpretation von neu definierten LaTeX-Makros sowie eine Syntaxkontrolle. TeXstudio kann ebenfalls Unicode-Dateien verarbeiten, ist durch Scripte erweiterbar und beherrscht Code-Faltung.

TeXnicCenter ist ein etwas älterer freier Texteditor unter Windows. Integrierte Funktionen erleichtern unter anderem die Strukturierung, Formatierung und Texthervorhebung der Dokumente. Der LaTeX-Quelltext wird mit integrierter Syntaxhervorhebung angezeigt. TeXnicCenter ist mit anderen integrierten Entwicklungsumgebungen für andere Programmiersprachen vergleichbar. Es hält LaTeX-Elemente als Icons oder Tastenkombinationen bereit. Weiterhin wird eine Projektverwaltung ähnlich wie bei TeXstudio geboten, und externe Programme lassen sich leicht einbinden. Das Übersetzen der LaTeX-Quelltexte in DVI, PostScript oder PDF wird per Tastenkombination oder Icon gesteuert. Treten Fehler auf, so kann per Mausklick direkt in den Quelltext zum Fehler gesprungen werden.

1.3.2 DVI-Dateien

Der nächste Schritt nach der Texteingabe besteht darin, aus der LaTeX-Datei mit dem Kommando `latex name.tex` eine DVI-Datei zu erstellen (Kennung *`*.dvi`*). Dabei handelt es sich um eine Datei, in der alle Anweisungen für das Seitenlayout in einer drucker- bzw. device-unabhängigen Sprache angegeben werden.

Bei der Ausführung von `latex` kommt es häufig zu Fehlermeldungen, die auf Syntaxfehler in der LaTeX-Datei zurückzuführen sind. Einige Informationen zum Umgang mit Fehlermeldungen und zur Fehlersuche finden Sie auf Seite 12.

Sobald die DVI-Datei vorliegt, kann sie mit den Programmen `xdvi` oder `kdvi` betrachtet werden. Wenn Sie die Datei ausdrucken möchten, ist noch ein weiterer Arbeitsschritt erforderlich – die Umwandlung der DVI-Datei in das PostScript-Format. Für diese Umwandlung ist das Programm `dvips` zuständig (Seite 17).

1.3.3 Postscript

PostScript-Dateien können je nach Distribution mit `ghostview`, `gv`, `ggv` oder `kghostview` am Bildschirm betrachtet werden. Falls Sie Ihren Drucker korrekt konfiguriert haben, können Sie die PostScript-Datei natürlich auch ausdrucken. Sie können aber auch weitermachen und eine PDF-Datei erzeugen. Im Einzelnen sieht der Weg von der Textdatei *`test.tex`* im LaTeX-Format bis hin zum Ausdruck folgendermaßen aus:

```
user$ latex test.tex                    # liefert test.dvi
user$ dvips -o test.ps test.dvi       # liefert test.ps
user$ ghostview test.ps &              # Ergebnis überprüfen
user$ lpr test.ps                       # Ausdruck (oder auch .pdf generieren)
```

Beim ersten Ausführen von `dvips` oder `xdvi` werden neue Schriftartdateien generiert. Die beiden Kommandos starten dazu automatisch das Programm `mf` (Metafont). Gegebenenfalls muss auch der `latex`-Durchlauf zweimal (in seltenen Fällen dreimal) erfolgen, damit alle Querverweise korrekt abgesättigt werden.

1.3.4 Compilierung automatisieren

Programmierprofis basteln sich meist ein *`Makefile`*, das den Anlauf automatisiert, zum Beispiel:

```
BUCH = linux

all:
     @echo "Aufruf : make <Option>"
     @echo
```

```
      @echo "Gueltige Optionen sind :"
      @echo "dvi   - Erzeugt Latex-Lauf und Anzeige ueber xdvi"
      @echo "ps    - Latex-Lauf und anschliessend dvips"
      @echo

dvi:
      @rm -f ${BUCH}.log missfont.log
      @latex ${BUCH}
      @makeindex -s myindex.sty ${BUCH}
      @latex ${BUCH}
      @makeindex -s myindex.sty ${BUCH}
      @xdvi ${BUCH}

ps:
      @rm -f ${BUCH}.log missfont.log
      @latex ${BUCH}
      @makeindex -s myindex.sty ${BUCH}
      @latex ${BUCH}
      @makeindex -s myindex.sty ${BUCH}
      @dvips -t a4 -o ${BUCH}.ps ${BUCH}.dvi
     @ghostview test.ps

coffee:
      @echo
      @echo "With milk and sugar?"
```

Wer keine Make-Dateien mag, kann auch ein Shell-Programm mit gleicher Funktion schreiben. Generell gilt, dass man sich nicht erst mit den Anfängen von LaTeX befassen sollte, wenn der Abgabetermin für Artikel, Diplomarbeit oder was auch immer schon in greifbare Nähe gerückt ist.

1.3.5 Fehlersuche in LaTeX-Texten

Der erste Kontakt mit dem Programm LaTeX ist in der Regel frustrierend. Das Programm arbeitet interaktiv, reagiert auf den ersten auftretenden Fehler mit einer fast immer unverständlichen Fehlermeldung und erwartet dann auch noch von Ihnen, dass Sie durch die Eingabe eines Buchstabens angeben, wie es weitergehen soll. Eine typische Fehlermeldung sieht beispielsweise so aus:

```
LaTeX error.  See LaTeX manual for explanation.
              Type  H <return>  for immediate help.
! Text for \verb command ended by end of line.
\@latexerr ...rcontextlines \m@ne \errmessage {#1}
                                                  \endgroup
l.46 ...das folgende Kommando {\verb?{\small
?
```

Die Fehlerursache ist in diesem Fall ein `\verb`-Kommando in Zeile 46, dessen Wirkung über das Ende dieser Zeile hinausreicht (und das ist nicht erlaubt). Sie haben jetzt folgende Möglichkeiten, LaTeX fortzusetzen:

- (↩) setzt die Verarbeitung der LaTeX-Datei ohne Rücksicht auf den gerade aufgetretenen Fehler fort. Manchmal funktioniert das, sehr häufig führt es zu zahlreichen Folgefehlern (auf die ebenfalls mit (↩) reagiert werden kann).
- (H) (↩) zeigt zusätzliche Informationen zur Fehlermeldung an. Der Infotext ist allerdings nur in den seltensten Fällen eine echte Hilfe.
- (R) (↩) setzt die Verarbeitung fort, zeigt weitere Fehlermeldungen an, erwartet aber keine Eingaben mehr.
- (Q) (↩) wie oben, aber ohne die Anzeige von Fehlermeldungen
- (X) (↩) beendet LaTeX.

In der Regel werden Sie LaTeX entweder mit (X) sofort beenden, wenn Sie die Fehlerursache erkannt haben, oder die Bearbeitung mit (Q) im Quiet-Modus fortsetzen. Im zweiten Fall können Sie darauf hoffen, dass LaTeX trotz der höchstwahrscheinlich auftretenden Folgefehler in der Lage ist, zumindest die beanstandete Seite fertig zu übersetzen. In diesem Fall können Sie mit `xdvi` das Ergebnis (die gesetzte Seite) ansehen und dort vielleicht die Fehlerursache erkennen.

In jedem Fall werden Sie anschließend in den Editor wechseln und dort den Fehler in der LaTeX-Datei suchen. Dabei ist die Datei *name.log* eine wesentliche Hilfe. In dieser Datei werden alle Fehlermeldungen von LaTeX gespeichert (unabhängig davon, ob sie auf dem Bildschirm angezeigt wurden oder nicht). Der Dateiname dieser Protokolldatei setzt sich aus dem Namen der übersetzten LaTeX-Datei und der Kennung *.log* zusammen. Die wichtigste Information in dieser Datei ist in der Regel die Nummer der Zeile, in der der Fehler aufgetreten ist.

Wenn Sie Probleme beim Aufspüren eines Fehlers haben, sollten Sie versuchen, den kritischen Textausschnitt zu isolieren und in eine eigene, möglichst kleine Datei zu kopieren. Generell empfiehlt sich bei umfangreichen Texten eine Zerlegung in mehrere Dateien.

LaTeX liefert während der Bearbeitung von Texten nicht nur Fehlermeldungen, sondern auch Warnungen. Bei Warnungen wird die Bearbeitung des Textes nicht unterbrochen. Viele Warnungen beginnen zumeist mit einem Text wie *overfull hbox* und deuten darauf hin, dass LaTeX Probleme beim Zeilen- oder Seitenumbruch hat. In solchen Fällen sind zumeist manuelle Eingriffe im Text erforderlich (Trennvorschläge, erzwungene Seitenumbrüche etc.). Oft ist LaTeX hierbei aber auch zu pingelig.

Abschließend einige Tipps und Hinweise zum richtigen Umgang mit LaTeX, wenn Probleme auftreten:

- Das interaktive Verhalten von LaTeX – dass also bei jedem Fehler eine Unterbrechung auftritt – kann sehr lästig sein, vor allem, wenn die Übersetzung automatisiert werden soll. Wenn Sie am Beginn der LaTeX-Datei die Anweisung `batchmode` einfügen, arbeitet LaTeX auch bei Fehlern interaktiv (so, als würde beim ersten Fehler (Q) eingegeben). Nach dem Ende der Übersetzung können Sie in Ruhe die **.log*-Datei lesen.
- Während LaTeX eine Tastatureingabe erwartet, reagiert das Programm nicht auf (Strg)+(C)! Wenn Sie das Programm während einer Eingabe beenden möchten, müssen Sie (Strg)+(D) (für End of File) drücken! Dieser Notausstieg ist insbesondere dann praktisch, wenn LaTeX auf einen falschen Dateinamen gestoßen ist und von Ihnen die Angabe einer anderen Datei erwartet. Einen alternativen Ausweg stellt die Eingabe von *null* dar. LaTeX lädt dann die leere Datei *null.tex* bzw. *null.sty*, die extra für diesen Zweck in der LaTeX-Verzeichnisstruktur vorgesehen ist.
- In seltenen Fällen steckt der Fehler nicht in der zu übersetzenden LaTeX-Datei, sondern in der Datei für das Inhaltsverzeichnis, die beim vorangegangenen Durchlauf erzeugt worden ist. Löschen Sie die Datei *name.toc*!
- Wenn bei der Übersetzung einer fremden **.tex*-Datei zahlreiche unerklärliche Fehlermeldungen auftreten, dann liegt das zumeist daran, dass es sich nicht um eine LaTeX-, sondern um eine TeX-Datei handelt. TeX-Dateien weisen ebenfalls die Kennung **.tex* auf, müssen aber mit `tex dateiname.tex` übersetzt werden!

Ein praktisches Hilfsmittel bei der Fehlersuche ist das Programm `lacheck`. Es analysiert die als Parameter übergebene LaTeX-Datei und liefert eine Liste mit Warnungen über mögliche Fehler.

1.3.6 LaTeX-Dateien suchen

Angesichts der unüberschaubaren Anzahl von Dateien kann die Suche nach einer bestimmten Font- oder Style-Datei relativ lange dauern. Bei der teTeX-Distribution wird daher eine zusätzliche Datenbank verwaltet, die eine Liste aller TeX-Dateien enthält. Die Datenbank besteht im Wesentlichen aus dem Ergebnis von `ls -R` und hat daher den Dateinamen *ls-R*.

Falls Sie teTeX manuell um zusätzliche Dateien erweitern, müssen Sie unbedingt das Kommando `texhash` ausführen, damit die Datenbank *ls-R* aktualisiert wird. Auch sonst sollte dieses Kommando die erste Maßnahme sein, wenn es Probleme beim Auffinden von TeX-Dateien gibt. Intern ist die Kpathsea-Library für die Suche nach TeX-Dateien zuständig. Wenn die Details Sie interessieren, lesen Sie *kpathsea.dvi* oder die äquivalenten `info`-Texte sowie die sehr kurzen `man`-Pages zu `texhash` und `ls-R`.

1.4 Einführungsbeispiel

Bevor der nächste Abschnitt eine systematische Beschreibung der wichtigsten LaTeX-Kommandos liefert, soll das folgende Beispiel den prinzipiellen Umgang mit LaTeX demonstrieren. Die unten abgedruckten LaTeX-Anweisungen ergeben nach der Übersetzung durch LaTeX die in der Abbildung dargestellte Seite. Bei einem längeren Artikel wäre es natürlich sinnvoll, das Inhaltsverzeichnis auf einer eigenen Seite darzustellen und dem ganzen Artikel eine Titelseite voran zu stellen – darauf wurde hier aus Platzgründen verzichtet.

```
% Dokumenttyp: zweispaltiger Artikel
\documentclass[a4paper,twocolumn,11pt]{article}
\usepackage{ngerman}             % deutsche Überschriften
\usepackage[latin1]{inputenc}    % Latin-1-Zeichensatz
\parindent 0pt                   % kein Einrücken der ersten Zeile
\parskip1ex                      % Leerraum zw. Absätzen
\columnsep 1cm                   % 1 cm Abstand zwischen den Spalten
\begin{document}                 % Beginn des eigentlichen Textes
\tableofcontents                 % Inhaltsverzeichnis einfügen

\section{\LaTeX-Einführung}

\section{Gestaltung des Schriftbilds}

Dieser Text zeigt einige Gestaltungsmöglichkeiten in \LaTeX:
\textbf{fette Schrift}, \textit{kursive Schrift},
\textsc{Kapitälchen}, \textsf{Sans Serif}, \texttt{Typewriter}. Neu in
\LaTeXe\ ist die Tatsache, dass sich Schriftattribute jetzt weitgehend
problemlos kombinieren lassen -- beispielsweise \textbf{\textit{fett
und kursiv}}. Natürlich kann auch die Schriftgröße verändert werden
von {\tiny ganz winzig} über {\small klein} bis {\Large ziemlich
groß}.

\section{Textblöcke und Rahmen}

{\small
\begin{minipage}[t]{4cm}
Mit der {\verb?minipage?}-Um\-ge\-bung können Textblöcke
nebeneinander angeordnet werden.
\end{minipage}
\hfill
\begin{minipage}[t]{3cm}
Das ist die zweite, etwas schmalere Minipage.
\end{minipage}
}

\hbox{}\hfill\fbox{
\begin{minipage}{5cm}
Hier wurde eine 5 cm breite Mini\-page durch ein vor- und ein
nachgestelltes {\small\tt \textbackslash{}hfill}-Kom\-man\-do zentriert und mit
{\small\tt \textbackslash{}fbox} eingerahmt.
\end{minipage}
}\hfill\hbox{}

\section{Aufzählungen}

\LaTeX\ hat viele Vorteile gegenüber anderen Programmen:

\begin{itemize}
\item Die Qualität der Ergebnisse spricht für sich.
\item Die Verarbeitungsgeschwindigkeit ist sehr hoch, wenn man sich
      an die Syntax gewöhnt hat.
\item \LaTeX-Texte sind portabel und werden in der Unix-Welt oft
      zur Online-Dokumentation eingesetzt.
\end{itemize}

\section{Fußnoten}

Dieser Absatz liefert zwei Beispiele für Fuß\-noten.\footnote{Das ist
die erste Fußnote.} \LaTeX\ nummeriert die Fuß\-noten\footnote{Die
zweite Fußnote.} natürlich automatisch.

\section{Mathematische Formeln}
```

```
Seine noch immer große Bedeutung verdankt \LaTeX\ in erster Linie
seinem hervorragenden Formelsatz.  Versuchen Sie, die folgenden
Formeln einmal in einem anderen Programm einzugeben! Formeln können
übrigens auch direkt im Text (etwa hier: $\pi \, \int x^2 dx$)
verwendet werden.

\[\lim _{n\rightarrow \infty }\sum _{k=1}^{n}\sqrt
  [3]{1+{\frac {k^{2}}{n^{3}}}}-n\]

\[\left [\begin {array}{cc}
{\frac {\partial }{\partial x}}f&{\frac {\partial }{\partial y}}f\\
\noalign{\medskip}
{\frac {\partial }{\partial x}}g&{\frac {\partial }{\partial y}}g
\end {array}\right ]\]
\end{document}
```

Wenn Sie nach dem Studium dieses Abschnitts den Eindruck gewonnen haben, dass die Arbeit mit LaTeX eine manchmal etwas mühselige Angelegenheit ist, haben Sie prinzipiell Recht. Der Weg bis zu einem wirklich optimalen Ergebnis ist oft dornig. Zum Teil sind manuelle Eingriffe notwendig, damit LaTeX deutsche Wörter richtig trennt und Seiten dort umbricht, wo es aus ästhetischen Gründen sinnvoll ist. Jedoch spricht die Qualität von LaTeX-Texten für sich.

Im mathematisch-naturwissenschaftlichen Sektor gibt es zudem keine ernsthafte Alternative zu LaTeX. Textverarbeitungsprogramme wie Microsoft Word oder OpenOffice Writer sind für umfangreiche wissenschaftliche Texte mit Formeln und Bildern zu ineffizient und oft auch zu instabil. Was nützt eine tolle Benutzeroberfläche, wenn ein Textverarbeitungsprogramm bei längeren Texten plötzlich abstürzt, Querverweise nicht mehr findet, statt Abbildungen nur noch ein rotes „X“ anzeigt? Der Einstieg in LaTeX dauert sicherlich ein paar Tage länger als bei anderen Programmen – aber diese Zeit holen Sie später wieder auf, wenn das Programm auch mit hundert- oder tausendseitigen Dokumenten noch problemlos funktioniert! Wie auch bei anderen Programmen können Sie sich für LaTeX Formatvorlagen erstellen, die dann nur noch ausgefüllt werden müssen. Berühmt ist es jedoch für seine Erweiterbarkeit. Es lassen sich jederzeit neue Makros oder komplette Pakete erstellen, die dann die Arbeit erleichtern. So handelt es sich beispielsweise bei den Tastensymbolen in diesem Buch um ein Makro *keys*, das mit dem Aufruf `\keys{Drück mich!}` das Ergebnis (Drück mich!) hat. So lassen sich auch Kästen und andere Elemente austüfteln und dann sehr einfach verwenden.

Inhaltsverzeichnis

1 LaTeX-Einführung

1.1 Gestaltung des Schriftbilds

Dieser Text zeigt einige Gestaltungsmöglichkeiten in LaTeX: **fette Schrift**, *kursive Schrift*, KAPITÄLCHEN, Sans Serif, `Typewriter`. Neu in LaTeX 2ε ist die Tatsache, dass sich Schriftattribute jetzt weitgehend problemlos kombinieren lassen – beispielsweise ***fett und kursiv***. Natürlich kann auch die Schriftgröße verändert werden von ganz winzig über klein bis ziemlich groß.

1.2 Textblöcke und Rahmen

Mit der `minipage`-Umgebung können Textblöcke nebeneinander angeordnet werden.

Hier wurde eine 5 cm breite Minipage durch ein vor- und ein nachgestelltes `\hfill`-Kommando zentriert und mit `\fbox` eingerahmt.

Das ist die zweite, etwas schmalere Minipage.

1.3 Aufzählungen

LaTeX hat viele Vorteile gegenüber anderen Programmen:

- Die Qualität der Ergebnisse spricht für sich.
- Die Verarbeitungsgeschwindigkeit ist sehr hoch, wenn man sich an die Syntax gewöhnt hat.
- LaTeX-Texte sind portabel und werden in der Unix-Welt oft zur Online-Dokumentation eingesetzt.

1.4 Fußnoten

Dieser Absatz liefert zwei Beispiele für Fußnoten.[1] LaTeX nummeriert die Fußnoten[2] natürlich automatisch.

1.5 Mathematische Formeln

Seine noch immer große Bedeutung verdankt LaTeX in erster Linie seinem hervorragenden Formelsatz. Versuchen Sie, die folgenden Formeln einmal in einem anderen Programm einzugeben! Formeln können übrigens auch direkt im Text (etwa hier: $\pi \int x^2 dx$) verwendet werden.

$$\lim_{n\to\infty} \sum_{k=1}^{n} \sqrt[3]{1+\frac{k^2}{n^3}} - n$$

$$\begin{bmatrix} \frac{\partial}{\partial x} f & \frac{\partial}{\partial y} f \\ \frac{\partial}{\partial x} g & \frac{\partial}{\partial y} g \end{bmatrix}$$

[1] Das ist die erste Fußnote.
[2] Die zweite Fußnote.

Bild 1.2: Der von LaTeX bearbeitete Text *article.tex*

2

LaTeX-Dokumente anzeigen und weiterverarbeiten

2.1 DVI-Dateien anzeigen (xdvi, kdvi)

Das Kommando `latex` erzeugt aus Ihrer LaTeX-Datei eine DVI-Datei. Mit den Programmen `xdvi` und `kdvi` können Sie *`*.dvi`*-Dateien auf dem Bildschirm ansehen und darin blättern. Denken Sie daran, dass einige PostScript-spezifische Gestaltungsmöglichkeiten in `xdvi` und `kdvi` nicht angezeigt werden können. Um ein LaTeX-Dokument exakt so zu sehen, wie es ausgedruckt wird, müssen Sie es in das PostScript-Format umwandeln und mit `ghostview`, `gv`, `ggv` oder `kghostview` betrachten.

Wenn das LaTeX-Dokument die LaTeX-eigenen Schriften verwendet, müssen deren Bitmaps bei der ersten Verwendung erzeugt werden. Dazu starten `xdvi` bzw. `kdvi` automatisch das LaTeX-Zusatzprogramm `metafont`. Deswegen kann das erstmalige Anzeigen einer DVI-Datei relativ lange dauern.

`xdvi` wird mit dem Dateinamen der *`*.dvi`*-Datei als Parameter gestartet. Anschließend kann mit den Buttons *Next*, *Previous* etc. in dem Dokument geblättert werden. Wenn Sie die Maus über den Bildausschnitt bewegen und dabei die linke Taste drücken, wird (verzögerungsfrei) ein vergrößerter Ausschnitt des Rechtecks unter der Maus dargestellt. Die folgende Tabelle enthält eine Übersicht der wichtigsten Tastenkürzel zur Steuerung von `xdvi`.

xdvi-Tastenkürzel

Bild ↑, Bild ↓	vorige/nächste Seite
Leertaste, ↩	nächste Seite
Backspace, Entf	vorige Seite
↑, ↓, ←, →	Bildausschnitt verschieben
G	springt zur vorher eingegebenen Seitennummer
V	(de)aktiviert die Anzeige von PostScript-Grafiken

`kdvi` muss bei manchen Distributionen extra installiert werden. Bemerkenswert ist die komfortable Export-Funktion für die Formate PostScript und PDF, die den manuellen Aufruf von `dvips` bzw. `dvipdfm` erspart.

2.2 PostScript-Dokumente erzeugen (dvips)

`dvips` wandelt *`*.dvi`*-Dateien in das PostScript-Format um. Die Syntax des Kommandos sieht so aus:

```
user$ dvips [optionen] -o name.ps name.dvi
```

-A wandelt nur ungerade Seiten um.

-B wandelt nur gerade Seiten um.

-D *n* verwendet bei der Erzeugung von LaTeX-Bitmap-Schriften eine Auflösung von *n* dpi (*dots per inch*). Die Standardauflösung beträgt meist 600 dpi. Alternativ darf *n* auch 300, 400 oder 1270 (für die Druckerei) betragen. Die Option ist nur für Bitmap-Schriften relevant. Eingesetzte PostScript-Schriften sind immer auflösungsunabhängig.

-E erzeugt eine EPS-Datei (*Encapsulated PostScript*) mit einer BoundingBox, die nur den tatsächlich genutzten Teil der Seite umfasst. Das ist nur sinnvoll, wenn die DVI-Datei nur eine Seite hat und die resultierende EPS-Datei anschließend in ein anderes Dokument eingebettet werden soll.

-G0 verhindert Inkompatibilitäten mit dem Adobe Reader. Diese Option ist nur zweckmäßig, wenn die PostScript-Datei später in eine PDF-Datei umgewandelt werden soll.

-i -S *n* zerlegt die Ausgabe in Dateien zu je *n* Seiten. Die Dateien werden automatisch durchnummeriert.

-l *letzteseite* beendet die Umwandlung mit der angegebenen Seite.

-o *zieldatei* schreibt das Ergebnis in die angegebene Datei (anstatt es an das Programm `lpr` weiterzuleiten).

-p *ersteseite* beginnt die Umwandlung mit der angegebenen Seite.

-pp *n1,n2-n3,n4,n5,n6-n7* druckt die angegebenen Seiten. Beachten Sie, dass in der Seitenliste keine Leerzeichen vorkommen dürfen.

-P*name* berücksichtigt zusätzliche `dvips`-Default-Einstellungen. Beispielsweise bewirkt `-Ppdf`, dass die resultierende Datei für eine spätere Umwandlung in ein PDF-Dokument optimiert wird (Konfigurationsdatei */var/lib/texmf/dvips/config/config.pdf*).

Das Kommando kann auch einfach als `dvips name` ausgeführt werden. Es liest dann *name.dvi* und schreibt das Ergebnis in *name.ps*. Globale Default-Einstellungen für *dvips* sind in */etc/texmf/config.ps* bzw. in */usr/share/texmf/dvips/config.ps* definiert.

Probleme mit pixeligen Schriften: In der Vergangenheit sahen aus LaTeX-Dokumenten erzeugte PostScript- und PDF-Dokumente oft pixelig aus. Der Grund besteht darin, dass die LaTeX-Originalschriften tatsächlich Bitmap-Schriften sind. Bei aktuellen teTeX-Versionen sollte dieses Problem nicht mehr auftreten, weil es nun auch zu den LaTeX-Originalschriften PostScript-Varianten gibt, die per Default eingesetzt werden. Verantwortlich dafür ist die Datei */usr/share/texmf/dvips/tetex/bsr.map*, die von `dvips` automatisch berücksichtigt wird.

Wenn Sie bei Ihrer LaTeX-Version dennoch mit diesem Problem kämpfen, gibt es folgende Lösungswege:

- Stellen Sie das gesamte Dokument auf PostScript-Schriften um (siehe Seite 88). Dazu reichen im Regelfall zwei oder drei `\usepackage`-Zeilen. Allerdings ändert sich durch diese Maßnahme der Zeilen- und Seitenumbruch.
- Falls Sie bei den LaTeX-Standardschriften bleiben möchten, entfernen Sie die Anweisung `\usepackage[T1]{fontenc}`. Momentan stehen PostScript-Varianten für die LaTeX-Standardschriften nur in der LaTeX-Font-Kodierung (CM-Schriften) zur Verfügung, nicht aber für die T1- und TS1-Kodierung (EC- und TC-Schriften). Hintergrundinformationen zu den CM-, EC- und TC-Schriften sowie zur Font-Codierung finden Sie auf Seite 87.
- Fügen Sie die folgende Zeile in *config.ps* ein:

  ```
  p +bsr.map
  ```

- Erhöhen Sie die Auflösung der Bitmap-Schriften mit der `dvips`-Option `-D`. Die Druckqualität steigt dadurch deutlich an. (Die Darstellung im Adobe Reader bleibt leider dennoch spürbar pixelig; es ist aber ein erstklassiger Ausdruck möglich.)

2.3 PDF-Dokumente erzeugen

Oft möchte man LaTeX-Dokumente als PDF-Datei weitergeben. Dazu gibt es eine ganze Menge Möglichkeiten. In allen Fällen erhalten Sie als Ergebnis eine PDF-Datei, die wie die äquivalente PostScript-Datei aussieht. Ob auch PDF-Zusatzfunktionen (Inhaltsverzeichnis, anklickbare Links etc.) genutzt werden können, hängt vom beschrittenen Umwandlungsweg und von den im LaTeX-Dokument eingesetzten Zusatzpaketen ab.

- Sie erzeugen zuerst mit `dvips` eine PostScript-Datei und wandeln diese dann mit `ps2pdf` oder mit dem Adobe Distiller in eine PDF-Datei um. (Adobe Distiller ist Teil des kommerziellen Programmpakets Adobe Acrobat, von dem es zurzeit leider keine Linux-Version gibt.) PDF-Funktionen können durch das LaTeX-Paket *hyperref* genutzt werden.
- Sie wandeln die DVI-Datei mit `dvipdfm` in eine PDF-Datei um. Für PDF-Funktionen müssen Sie zusätzliche `\special`-Kommandos in das LaTeX-Dokument einfügen und *hyperref* nutzen.
- Sie wandeln die LaTeX-Datei mit `pdflatex` direkt in eine PDF-Datei um (wie schon im Abschnitt 9 beschrieben). Dieses Programm sieht eine Reihe zusätzlicher LaTeX-Kommandos vor, um die PDF-Funktionen zu steuern.

dvipdfm fasst einige Teilschritte in einem Programm zusammen und kann einfach als `dvipdfm name` ausgeführt werden. Es liest dann *`name.dvi`* und schreibt das Ergebnis in *`name.pdf`*. Das Programm `dvipdfm` ist vor allem dann interessant, wenn die in der Dokumentation (Datei *`dvipdfm.pdf`*) beschriebenen `\special`-Kommandos eingesetzt werden:

```
user$ dvipdfm [options] name.dvi $¿$ name.pdf
```

-l verwendet das Querformat (*landscape*).

-p *papersize* verwendet das angegebene Papierformat (z. B. letter, legal, a3, a4 oder a5).

-r *dpi* verwendet den angegebenen dpi-Wert bei der Erzeugung von Bitmap-Fonts der LaTeX-Schriften.

-s *pages* gibt die gewünschten Seiten an (z. B. `1,3,7,9-12`).

-z *n* gibt den gewünschten Kompressionsgrad (0-9) an. 9 bedeutet maximale Kompression.

pdflatex ist eine Variante zu LaTeX, die speziell dafür entwickelt wurde, PDF-Dateien zu erzeugen. Dementsprechend gibt es eine Reihe zusätzlicher Kommandos. Die größte Einschränkung besteht darin, dass `pdflatex` nicht mit allen LaTeX-Erweiterungen zurechtkommt. Das folgende Kommando liefert als Ergebnis direkt die PDF-Datei *`name.pdf`*:

```
user$ pdflatex name.tex
```

Weitere Informationen und Beispiele finden Sie in den Dateien des Verzeichnisses `/usr/share/texmf/doc/pdftex/base`. Die zentrale Anlaufstelle im Internet finden Sie unter *http://www.tug.org/applications/pdftex/*.

hyperref: Das `hyperref`-Paket hilft, die Möglichkeiten des PDF-Formats besser zu nutzen. Die folgenden Zeilen am Beginn eines LaTeX-Dokuments bewirken, dass das PDF-Dokument mit einem PDF-kompatiblen Inhaltsverzeichnis ausgestattet wird (ausklappbare Bookmarks) und alle Querverweise innerhalb des Dokuments blau hervorgehoben werden. Die Querverweise können per Maus angeklickt werden.

```
\usepackage[ps2pdf]{hyperref}
\hypersetup{colorlinks=true, linkcolor=darkblue, urlcolor=blue}
```

Querverweise in das Internet können mit `\url{http://www.adresse.com}` oder mit `\href{http://adresse}{beschreibender Text}` formuliert werden und sind dann ebenfalls anklickbar.

Das `hyperref`-Paket ist mit `dvipdfm` inkompatibel. Die Funktionen des `hyperref`-Pakets sind wirksam, wenn Sie die PDF-Datei mit `pdflatex` `dvipdf` oder mit `dvips` und `ps2pdf` erzeugen!

`hyperef` bietet noch viel mehr Zusatzfunktionen, die unter anderem im Buch *Mit LaTeX ins Web* von Michael Goossens und Sebastian Rahtz dokumentiert sind. Einen exzellenten Überblick über die zahlreichen Möglichkeiten, LaTeX-Texte in das PDF-Format umzuwandeln, gibt das PDF-Dokument *http://www.ctan.org/tex-archive/info/german/LaTeX2PDF.pdf*.

2.4 HTML-Dokumente erzeugen

Wenn Sie HTML-Dateien aus LaTeX-Dokumenten erstellen möchten, bieten sich zwei Werkzeuge an: `latex2html` und `tex4ht`.

Bei **latex2html** handelt es sich um ein Perl-Script, das aus einer LaTeX-Datei eine oder mehrere *`*.html`*-Dateien erzeugt. Formeln und LaTeX-Sonderzeichen werden in *`*.gif`*-Bilder übersetzt. Dazu werden das Grafikpaket `netpbm` sowie `gs` eingesetzt.

`latex2html` zerlegt das Dokument in einzelne Abschnitte und speichert alle resultierenden HTML- und PNG-Dateien im neuen Verzeichnis *`name`*. Das Programm funktioniert nur dann wunschgemäß, wenn nur LaTeX-Standardkommandos eingesetzt werden (nicht aber diverse eigene Makros, Zusatzpakete etc.). Meist funktioniert die Übersetzung nur bei einfacheren Texten. Weitere Informationen zu `latex2html` finden Sie unter *http://www.latex2html.org/*.

tex4ht muss in das LaTeX-Dokument mittels `\usepackage{tex4ht}` eingefügt werden. Anschließend wird der Text mit dem Kommando `ht latex [options] name.tex` übersetzt. Als Resultat erhalten Sie die Datei *`name.html`* und eventuell eine Reihe weiterer HTML-Dateien. Als Basis für die Umwandlung in das HTML-Format dient eine DVI-Datei. `tex4ht` ist ausführlich im oben erwähnten Buch *Mit LaTeX ins Web* beschrieben. Einen Überblick über die Konfigurationsmöglichkeiten gibt die Webseite *http://www.cis.ohio-state.edu/ xgurari/TeX4ht/*.

3

Elementare LaTeX-Kommandos

Dieser Abschnitt fasst die wichtigsten LaTeX-Kommandos zusammen. Diese Kommandos sollten ausreichen, um einfache LaTeX-Dokumente zu erstellen. Wenn Sie intensiver mit LaTeX arbeiten möchten, führt aber an weiterführender Literatur kein Weg vorbei.

3.1 Formale Details

Die Arbeit mit LaTeX beginnt in einem beliebigen Texteditor. Im Text gilt eine Gruppe zusammenhängender Zeilen als Absatz. Wörter werden durch Leerzeichen oder Zeilenumbrüche voneinander getrennt, Absätze durch mindestens eine Leerzeile. Der eigentliche Zeilenumbruch innerhalb eines Absatzes wird von LaTeX durchgeführt. Deswegen spielt es keine Rolle, an welcher Stelle im Ausgangstext eine neue Zeile begonnen wird.

LaTeX-Kommandos (genauer: der Aufruf eines LaTeX-Makros) beginnen immer mit einem Backslash \. Wenn die Kommandos Parameter benötigen, stehen diese in geschweiften Klammern, also beispielsweise `\chapter{Kapitelüberschrift}`. Manche Kommandos kennen auch optionale Parameter, die in eckigen Klammern angegeben werden, z. B. `$\sqrt[3]{x}$` für $\sqrt[3]{x}$. Ohne `[3]` liefert `\sqrt` eine gewöhnliche Quadratwurzel, also $\sqrt{x}$. Geschweifte Klammern können auch dazu verwendet werden, die Wirksamkeit von Kommandos einzuschränken. So wird durch `{\bf Wort}` nur ein einziges Wort fett gedruckt, während das ungeklammerte Kommando `\bf` das Attribut „Fett“ bis auf Widerruf einstellt. Manchmal dienen sie auch zur Trennung vom nachfolgenden Text, wenn sonst das Makro nicht erkannt würde (z. B. `\textbackslash{}LaTeX` statt `\textbackslashLaTeX`).

Umgebungen stellen einen besonderen Typ von Kommandos dar. Sie werden mit `\begin{umgebung}` eingeleitet und mit `\end{umgebung}` abgeschlossen. Für den gesamten Text zwischen diesen beiden Kommandos gelten die besonderen Formatierungsmerkmale der Umgebung. Typische Umgebungsnamen sind *tabbing* (für Tabellen) oder *verbatim* für Listings mit Sonderzeichen.

Innerhalb des LaTeX-Textes können mit % Kommentare eingeleitet werden. Der Rest der Zeile ab diesem Zeichen wird von LaTeX nicht beachtet.

3.2 Vorspann

LaTeX-Texte beginnen mit dem Kommando `\documentclass[optionen]{typ}`. Dieses Kommando bestimmt den Texttyp. LaTeX kennt in der Standardkonfiguration vier wichtige Texttypen: *book*, *report*, *article* und *letter*. Die drei ersten Texttypen sind einander relativ ähnlich und unterscheiden sich primär in der vorgesehenen Textlänge. *article* kennt im Gegensatz zu *book* und *report* keine Kapitel; die kleinste Gliederungseinheit ist dort ein Abschnitt (*section*). In *book* werden alle Seiten automatisch mit Kopfzeilen ausgestattet, in denen neben der Seitennummer auch der Name des aktuellen Kapitels (gerade Seiten) und der Name des aktuellen Abschnitts (ungerade Seiten) angegeben wird. Am

leichtesten erkennen Sie die Unterschiede zwischen den drei Texttypen *book*, *report* und *article*, wenn Sie die erste Zeile der Beispieldatei des vorangegangenen Abschnitts ändern und dort der Reihe nach alle drei Typen einsetzen.

Der Texttyp *letter* kann zum Verfassen von Briefen verwendet werden. Auf diesen Texttyp wird hier aus Platzgründen allerdings nicht eingegangen. LaTeX-intern werden Texttypen durch Makrodateien mit der Kennung `*.cls` realisiert. Diese Dateien sind im Verzeichnis `/usr/share/texmf/tex/latex/base` gespeichert. Manche Verlage, Universitäten etc. stellen darüber hinaus eigene Makrodateien zur Verfügung, die sich speziell zur Formatierung von wissenschaftlichen Artikeln, Diplomarbeiten etc. eignen.

Vor dem Texttyp können in eckigen Klammern Optionen angegeben werden. Wichtige Optionen sind `11pt` und `12pt` (sie verändern die Standardschriftgröße), *a4paper* (DIN-A4-Papierformat) und `twocolumn` (zweispaltige Texte).

LaTeX-Dokumentklassen

```
\documentclass[optionen]{typ}
```

Texttypen

`book`	für lange Texte (Bücher), Gliederung in Teile, Kapitel, Abschnitte
`report`	wie oben, aber für kürzere Texte; andere Titelseite etc.
`article`	für Artikel, Gliederung in Abschnitte, Unterabschnitte; im Gegensatz zu *book* und *report* keine Unterscheidung zwischen geraden und ungeraden Seitenzahlen
`letter`	für Briefe (Für uns besser geeignet ist `dinbrief`.)

Optionen

`11pt`	Standardschriftgröße 11 (statt 10) Punkt
`12pt`	Standardschriftgröße 12 (statt 10) Punkt
`a4paper`	DIN-A4-Format (statt US-Letter)
`twoside`	Unterscheidung gerade/ungerade Seite (Standard bei *book*)
`twocolumn`	zweispaltiger Text

Tipp

Statt *book*, *report*, *article* und *letter* können Sie auch *scrbook*, *scrreprt*, *scrartcl* bzw. *scrlttr2* verwenden. Diese Formatvorlagen sind besser an die DIN-Papierformate angepasst. Außerdem werden alle Überschriften in Grotesk-Schriften (Sans-Serif-Schriften) ausgeführt.
Diese Formatvorlagen sind Teil des KOMA-Pakets, das üblicherweise als Teil der teTeX-Distribution installiert wird. KOMA ist das Kürzel für Markus Kohm, der die Dateien zusammengestellt hat.

Natürlich kann man die Standardvorgaben für die Dokumentklassen jederzeit nach eigenen Wünschen ändern. LaTeX definiert jeden Teil einer Seite mit entsprechenden Werten (siehe Abbildung), die sich den aktuellen Gegebenheiten anpassen lassen. Das Setzen der neuen Werte erfolgt mittels `\setlength`, z. B.:

```
\setlength\paperwidth       {297mm}
\setlength\paperheight      {187mm}

\setlength\textheight       {198mm}
\setlength\textwidth        {126mm}
\setlength\marginparwidth   {32mm}
\setlength\marginparsep     {3.5mm}
\setlength\oddsidemargin    {4.6mm}
\setlength\evensidemargin   {4.6mm}
\setlength\topmargin        {-11.9mm}
```

Normalerweise kann man die Angaben aber so lassen, wie sie von LaTeX bzw. TeX vorgegeben worden sind.

Diverse LaTeX-Zusatzfunktionen können durch weitere Pakete aktiviert werden. Dabei handelt es sich um LaTeX-Dateien mit der Kennung `*.sty`, die diverse Systemeinstellungen

verändern und zusätzliche Kommandos zur Verfügung stellen. Zusatzpakete werden mit `\usepackage[optionen]{name}` geladen. Das bzw. die `\usepackage`-Kommandos müssen unmittelbar nach `\documentclass` angegeben werden. Die folgende Tabelle zählt nur die wichtigsten Zusatzpakete auf. (Der Platz in diesem Kapitel reicht leider nicht aus, um alle Pakete zu beschreiben.)

Zusatzpakete

`\usepackage{alltt}`	Variante zur `verbatim`-Umgebung
`\usepackage{babel}`	Mehrsprachige Dokumente
`\usepackage{color}`	Farben nutzen
`\usepackage{fancyhdr}`	Gestaltung der Kopf- und Fußzeilen
`\usepackage[T1]{fontenc}`	Andere Zeichenkodierung
`\usepackage{german}`	Deutsche Trennungen, Überschriften etc.
`\usepackage{graphicx}`	Einbindung von Grafikdateien
`\usepackage{hyperref}`	PDF- und HTML-Funktionen
`\usepackage[latin1]{inputenc}`	Latin-1-Zeichensatz
`\usepackage[utf8x]{inputenc}`	UTF-8-Zeichensatz verwenden
`\usepackage{makeidx}`	Stichwortverzeichnis
`\usepackage{ngerman}`	Wie `german`, aber neue Trennregeln
`\usepackage[sf]{titlesec}`	Sans-Serif-Überschriften

LaTeX erwartet den Quelltext per Default als ASCII-Datei. Wenn Sie in der LaTeX-Datei die westeuropäischen Sonderzeichen des Latin-1-Zeichensatzes verwenden möchten, müssen Sie am Beginn des LaTeX-Dokuments die folgende Zeile einfügen (das German-Paket auch gleich noch mit):

```
\usepackage[latin1]{inputenc}
\usepackage{ngerman}
```

Wenn Sie die LaTeX-Datei als Unicode-Datei (UTF-8) speichern, müssen Sie am Beginn des LaTeX-Dokuments die folgende Zeile einfügen. Bei manchen Distributionen funktioniert das allerdings erst nach der Installation eines Extrapakets (unter Fedora `tetex-unicode`).

```
\usepackage[utf8x]{inputenc}
```

Das Unicode-Paket muss oft eigens installiert werden. Wenn das Paket bei Ihrer Distribution fehlt, finden Sie es im Internet unter *http://www.unruh.de/DniQ/latex/unicode/*. Zukünftige LaTeX-Versionen werden eine UTF-8-Encoding-Datei direkt enthalten. Die Notwendigkeit des UCS-Pakets entfällt dann.

3.3 Maßangaben

Zahlenwerte in Kommandos erfolgen bei allen Maßangaben als Dezimalzahl. Sie können sowohl mit einem Dezimalpunkt als auch mit einem Komma angegeben werden. Der Zahlenwert muss immer mit einer Einheit abgeschlossen werden (auch bei 0). Die Angabe kann nicht nur absolut, sondern auch als Faktor einer bekannten Größe (z. B. `0.6 \textwidth` erfolgen. Als Einheiten sind folgende Werte möglich:

Maßeinheiten

sp	1 scaled point = 1/65536 pt
pt	1 point = 1/72,269 Zoll = 0,351 mm
bp	1 big point = 1/72 Zoll
dd	1 Didôt point = 1/72 französische Zoll = 0,376 mm
mm	1 Millimeter = 2,845 pt
pc	1 pica = 12 pt = 4,218 mm
cc	1 Cicero = 12 dd = 4,531 mm
cm	1 Centimeter = 2,371 pc
in	1 inch = 72,269 pt = 25,4 mm = 6,022 pc
ex	die Höhe eines x in der aktuellen Schriftart
em	die Breite eines Gedankenstrichs (–) in der aktuellen Schriftart

3.4 Strukturierung von Texten

Der eigentliche Text wird mit dem Kommando `\begin{document}` eingeleitet und mit `\end{document}` abgeschlossen. Der Text innerhalb dieser beiden Kommandos wird dann durch LaTeX verarbeitet und gesetzt. Dabei stehen einige Kommandos zur Verfügung, mit denen der Text strukturiert werden kann:

Dokumentstrukturierung	
`\part{Überschrift}`	Teil (nur für `book` und *report*)
`\chapter{Überschrift}`	Kapitel (nur für `book` und *report*)
`\section{Überschrift}`	Abschnitt
`\section{Überschrift}`	Unterabschnitt
`\subsection{Überschrift}`	Unterunterabschnitt

LaTeX kümmert sich selbstständig um die Nummerierung der Kapitel und Abschnitte. Gleichzeitig wählt LaTeX automatisch geeignete Schriftarten und Textabstände für die Überschriften. Beim Texttyp *book* werden die Texte des `\chapter`- und `\section`-Kommandos auch beim Erstellen der Kopfzeilen berücksichtigt. Wenn die Kommandos in der Form `\section[kurzfassung]{vollständig}` verwendet werden, wird die Kurzfassung für die Kopfzeile und das Inhaltsverzeichnis, die vollständige Variante dagegen unmittelbar im Text verwendet.

Falls der Text mit einem Anhang ausgestattet werden soll, wird dieser mit `\appendix` eingeleitet. Innerhalb des Anhangs können wieder `\chapter`, `\section` etc. verwendet werden, wobei Kapitel jetzt mit A, B, C ... nummeriert werden.

Ein typisches LaTeX-Dokument beginnt meist so ähnlich wie das folgende Beispiel:

```
\documentclass[a4paper,11pt]{article}  % Dokumententyp
\usepackage{ngerman}                   % Zusatzpakete
\usepackage[latin1]{inputenc}

% eventuell Inhaltsverzeichnis einfügen
\tableofcontents

%Anzahl der Warnungen (underfull/overfull) reduzieren (es bleiben genug!)
\sloppy

\begin{document}
% der eigentliche Text, strukturiert durch
% \section{...}, \section{...} etc.

\appendix
% der Text des Anhangs, wiederum strukturiert durch
% \section{...}, \section{...} etc.
\end{document}
```

3.5 Gestaltung des Schriftbilds

Von herkömmlichen Textverarbeitungsprogrammen sind Sie vermutlich gewohnt, dass Sie bei der Auswahl von Schriftarten und -größen praktisch unbeschränkt flexibel sind. In LaTeX ist das aus historischen Gründen nicht der Fall. Im Regelfall sind Sie in LaTeX auf drei Schriftfamilien beschränkt: eine Standardschrift, die für den Fließtext, die Überschriften etc. verwendet wird, die Schrift `Typewriter` für Programmlistings und die Schrift Sans Serif für Hervorhebungen und andere Aufgaben.

Diese drei Schriftfamilien können in unterschiedlichen Attributen formatiert werden. Die meisten Attribute existieren bei der Standardschrift, die fett, kursiv, geneigt und in Kapitälchen dargestellt werden kann.

Die Unterscheidung zwischen kursiv und geneigt wird Ihnen vermutlich unbekannt vorkommen: Bei geneigten Schriften verwendet LaTeX die normale Schrift und neigt die Buchstaben ein wenig nach rechts. Bei kursiven Schriften wird dagegen eine eigene Schriftart verwendet, in der die Zeichen stärker geneigt und auch ein wenig anders geformt sind. Diese Unterscheidung gilt allerdings nicht bei allen Schriftarten.

Standardschrift	normal	*kursiv*	*geneigt*	KAPITÄLCHEN
	fett	***fett kursiv***	***fett geneigt***	**FETT KAPITÄLCHEN**
Sans Serif	normal	*kursiv*	*geneigt*	KAPITÄLCHEN
	fett	***fett kursiv***	***fett geneigt***	**FETT KAPITÄLCHEN**
`Typewriter`	`normal`	*`kursiv`*	*`geneigt`*	KAPITÄLCHEN

Aus historischen Gründen gibt es mehrere alternative Kommandos zur Einstellung der Schriftfamilien und -attribute. In der folgenden Tabelle sind in den beiden ersten Spalten die offiziellen Kommandos von LaTeX 2ε, in der dritten Spalte die noch immer erlaubte alte Syntax und in der vierten Spalte das Resultat aufgelistet. Das Ergebnis der verschiedenen Kommandos ist nur scheinbar dasselbe (siehe die folgende Tabelle).

`\emph` schaltet nicht einfach auf kursive Schrift um, sondern wechselt das Schriftattribut zwischen normal und kursiv. Innerhalb einer kursiven Schrift liefert `\emph` daher eine normale Schrift.

Die Kurzkommandos wie `\sl`, `\bf` etc. wirken zwar wegen des minimalen Tippaufwands attraktiv, haben aber aus Kompatibilitätsgründen eine recht eigenwillige Funktionsweise: So wechseln `\bf`, `\it`, `\sl` und `\it` in die Standardschriftart und deaktivieren alle anderen Attribute! Es ist mit diesen Kommandos also nicht möglich, Schriftattribute zu kombinieren. Verwenden Sie unbedingt die neueren `\textxy`-Kommandos, etwa `\textbf{\textit{text}}`!

Schriftattribute

LaTeX 2ε		LaTeX2.09	Ergebnis
`\textrm{text}`	`{\rmfamily text}`	`{\rm text}`	Standard (Roman)
`\textsf{text}`	`{\sffamily text}`	`{\sf text}`	Sans Serif
`\texttt{text}`	`{\ttfamily text}`	`{\tt text}`	`Typewriter`
`\textbf{text}`	`{\bferies text}`	`{\bf text}`	**fett**
`\textmd{text}`	`{\mdseries text}`	`{\rm text}`	normal
`\textit{text}`	`{\itshape text}`	`{\it text}`	*kursiv (italic)*
`\textsl{text}`	`{\slshape text}`	`{\sl text}`	*geneigt (slanted)*
`\textsc{text}`	`{\itshape text}`	`{\it text}`	*Kapitälchen*
`\textup{text}`	`{\upshape text}`	`{\rm text}`	normal
`\emph{text}`		`{\em text}`	*hervorgehoben*

Auch bei den Schriftgrößen unterscheidet sich LaTeX von dem, was Sie von anderen Textverarbeitungsprogrammen gewohnt sind. Es gibt eine Standardschriftgröße, die für das gesamte Dokument gilt. Diese Schriftgröße kann nur zwischen 10 und 12 Punkt variieren. 10 Punkt ist die Default-Einstellung; auf 11 oder 12 Punkt können Sie durch die `\documentstyle`-Optionen `11pt` oder `12pt` umstellen.

Von dieser Standardschriftgröße ausgehend, berechnet LaTeX automatisch passende Schriftgrößen für Überschriften durch `\chapter` oder `\section`, für Fußnoten, mathematische Formeln etc. Gleichzeitig gilt diese Größe als Anhaltspunkt für die Kommandos zur Veränderung der Schriftgröße:

Schriftgröße

`\tiny`	nur noch mit der Lupe zu lesen	`\Large`	größer
`\scriptsize`	winzig	`\LARGE`	noch größer
`\footnotesize`	sehr klein	`\huge`	riesig
`\small`	klein	`\Huge`	kingsize
`\normalsize`	Standardschrift		
`\large`	groß		

Um Text hoch[zustellen], verwenden Sie `\textsuperscript{text}`. In mathematischen Formeln gilt dagegen die Schreibweise `$a^{b}$` für a^b bzw. `$a_{10}$` für a_{10}.

3.6 Sonderzeichen

Sehr viele Sonderzeichen wie % oder $ gelten in LaTeX als Kommandos. Einige weitere Sonderzeichen haben zwar nur im mathematischen Modus eine besondere Bedeutung (z. B. ^ und _), können aber im normalen Textmodus ebenfalls nicht verwendet werden. Die folgende Tabelle fasst die Bedeutung der wichtigsten Sonderzeichen zusammen:

Bedeutung von Sonderzeichen

`%`	leitet Kommentare ein
`~`	festes Leerzeichen (z. B. in `5~cm`)
`{..}`	klammert Textbereiche ein (z. B. für besondere Formatierung)
`$formel$`	klammert Formeln im Fließtext ein
`_`	tiefstellen (nur im mathematischen Modus)
`^`	hochstellen (nur im mathematischen Modus)

Der Versuch, Sonderzeichen im laufenden Text unverändert darzustellen, ist eine Quelle beständigen Ärgers: Zum einen wird es Ihnen auch nach monatelangem Arbeiten mit LaTeX noch passieren, dass Sie einfach übersehen, dass ein Zeichen eine besondere Bedeutung hat. Zum anderen fehlt eine einheitliche Methode, um Sonderzeichen im Text darzustellen. In vielen Fällen reicht es, wenn dem Sonderzeichen einfach ein Backslash vorangestellt wird (etwa `\%`, um ein %-Zeichen zu erzeugen). Wenn das nichts hilft, kann der Code des Zeichens direkt mit `\char`*n* angegeben werden. Bei den meisten Textzeichen gilt dabei der normale ASCII-Code.

Die folgende Tabelle fasst die Kommandos zur Darstellung der wichtigsten Sonderzeichen zusammen. Beachten Sie, dass es bei manchen Zeichen unterschiedliche Varianten gibt, die sich typografisch oft ein wenig unterscheiden.

Darstellung von Sonderzeichen

`--`	– (Gedankenstrich)	`\char34{}`	"
`---`	— (langer Strich)	`\textbackslash`	\
`\_`	_ (Unterstrich)	`{\tt<}`	<
`{\tt\char95}`	_ (Unterstrich)	`{\tt>}`	>
`\#`	#	`\textless`	<
`\$`	$	`\textgreater`	>
`\&`	&	`\textasciitilde`	~
`\%`	%	`\pounds`	£
`\{`	{	`\copyright`	©
`\}`	}	`"‘` („xxx...)	„
`\textbar`	|	`"’` (...xxx“)	“
`\textasciicircum`	^		

Bei der Anwendung der obigen Kommandos ist Vorsicht geboten: LaTeX eliminiert nach manchen Kommandos ein eventuell erwünschtes Leerzeichen. So wird aus `\copyright X. B. Liebig` „©X. B. Liebig“. Damit zwischen © und dem folgenden Text ein Leerzeichen angezeigt wird, muss das Kommando mit einem Backslash oder mit geschwungenen Klammern abgeschlossen werden, also `\copyright\ X. B. Liebig` oder `\copyright{} X. B. Liebig`.

Zahlreiche Sonderzeichen können sehr einfach im mathematischen Modus gebildet werden – etwa π mit `$\pi$`, $\rightarrow$ mit `$\rightarrow$`, $<$ mit `$<$` oder $\leftarrow$ mit `$\leftarrow$`. Beachten Sie aber, dass dabei eine andere Schrift verwendet wird. Eine Zusammenstellung der wichtigsten mathematischen Zeichen finden Sie ab Seite 48.

Einen alternativen Weg zur Darstellung von Texten mit Sonderzeichen bietet das Kommando `\verb_text_`. `\verb_ls | more_` liefert `ls | more`. In diesem Beispiel wurde der Unterstrich als Klammerzeichen für den eigentlichen Text verwendet. Sie können aber ebenso ein beliebiges anderes Zeichen verwenden, das im darzustellenden Text nicht vorkommt. Der Text innerhalb von `\verb` wird in der Schriftart `typewriter` dargestellt. `\verb` kann in zahlreichen Umgebungen und insbesondere zur Definition von Makros nicht verwendet werden. `\verb`-Texte sind auf maximal eine Zeile beschränkt. Es gibt keine Möglichkeit, `\verb`-Texte fett darzustellen.

Wenn Sie nicht nur einige Zeichen oder Wörter mit Sonderzeichen darstellen möchten, sondern mehrere Zeilen, bietet sich dazu die `verbatim`-Umgebung an. Diese Umgebung wird mit `\begin{verbatim}` eingeleitet und mit `\end{verbatim}` abgeschlossen. Alle dazwischen angegebenen Programmzeilen werden unverändert in der `typewriter`-Schriftart dargestellt. Die `verbatim`-Umgebung eignet sich insbesondere zur Wiedergabe von Programmlistings, in denen es zumeist von Sonderzeichen nur so wimmelt.

verbatim-Umgebung

`\verb_text_`	stellt den Text innerhalb der Delimiter inklusive aller Sonderzeichen dar (anstelle des Unterstrichs kann jedes beliebige Zeichen verwendet werden, das im Text nicht vorkommt
`\begin{verbatim}`	leitet Text ein, der unverändert ausgegeben wird
`Beispieltext mit Sonderzeichen $ % ^ ~`	
`\end{verbatim}`	Ende der `verbatim`-Umgebung

Statt der `verbatim`-Umgebung können Sie auch die `alltt`-Umgebung einsetzen, wenn Sie das gleichnamige Zusatzpaket aktiviert haben. Damit können Sie innerhalb der Umgebung die Schriftart durch LaTeX-Kommandos verändern. Allerdings werden die Zeichen {, }, |und \ nun als LaTeX-Zeichen interpretiert.

3.7 Akzente und besondere Buchstaben

Die meisten westeuropäischen Sonderzeichen (Latin-1) können direkt im Text angegeben werden, sofern das LaTeX-Dokument einen geeigneten Zeichensatz verwendet (z. B. `\usepackage[latin1]{inputenc}`, siehe Seite 23). Darüber hinaus bietet LaTeX aber auch die Möglichkeit, Buchstaben und Akzente beinahe beliebig zu kombinieren. Die folgende Tabelle zeigt die wichtigsten Kombinationsmöglichkeiten für den Buchstaben a.

Buchstaben und Akzente kombinieren

`\'a`	á	`\`a`	à	`\^a`	â	`\~a`	ã
`\.a`	ȧ	`\=a`	ā	`\b{a}`	a̲	`\d{a}`	ạ
`\u{a}`	ă	`\v{a}`	ǎ	`\c{a}`	ą	`\"{a}`	ä

Für einige weitere Buchstaben gibt es besondere Codes: `\ae`, `\AE`, `\aa`, `\AA`, `\oe`, `\OE`, `\o` und `\O` liefern æ, Æ, å, Å, œ, Œ, ø und Ø.

3.8 Euro-Symbol

LaTeX selbst kennt zwar kein Euro-Symbol, es gibt aber eine Reihe von Möglichkeiten, das Zeichen in LaTeX-Dokumenten einzufügen. Das Paket `textcomp` stellt ein Euro-Symbol mit dem Kommando `\texteuro` zur Verfügung: €

Das Paket `eurosym` stellt ein Euro-Symbol mit dem Kommando `\euro` zur Verfügung und zwar auch fett und kursiv. Eine weitere Variante besteht darin, selbst ein Euro-Symbol zu definieren, indem Sie die folgenden Zeilen an den Beginn des LaTeX-Dokuments stellen. `\myeuro` liefert nun das Symbol €, das allerdings nicht exakt dem offiziellen Symbol entspricht (Haben Sie es bemerkt? Unser erstes eigenes Makro!).

```
\newcommand\myeuro{{\sffamily C%
  \makebox[0pt][l]{\kern-.70em\mbox{--}}%
  \makebox[0pt][l]{\kern-.68em\raisebox{.25ex}{--}}}}
```

3.9 Tabulatoren

Mit LaTeX haben Sie mehrere Möglichkeiten zur Definition von Tabellen. Die einfachste Variante ist die `tabbing`-Umgebung, bei der Sie Tabulatoren setzen können. Die Syntax dieser Umgebung sieht folgendermaßen aus:

Tabulatoren

```
\begin{tabbing}
muster \= muster \= \kill     Musterzeile: \= definiert Tabulatoren
term1 \> term2 \> term3\\     Tabelle: \> für Tabulator, \\ für Zeilenende
term4 \> term5                letzte Zeile ohne \\
\end{tabbing}
```

Die Tabellen werden also mit einer Musterzeile eingeleitet. In dieser Musterzeile werden die Positionen der linksbündigen Tabulatoren mit `\=` festgelegt. Als Musterzeile verwenden Sie normalerweise die breiteste Zeile aus Ihrer Tabelle. In diese Zeile sollten Sie vor jedem `\=` einen zusätzlichen Leerraum einfügen, beispielsweise mit einer `\qquad`-Anweisung (Leerraum in der Größe zweier Gedankenstriche (siehe Seite 52). `\kill` löscht die Musterzeile, sodass diese Zeile nur als Muster verwendet, aber nicht ausgegeben wird. Dazu ein Beispiel: Die Sonderzeichentabelle von Seite 26 wurde mit den folgenden LaTeX-Anweisungen produziert:

```
\begin{tabbing}
{\verb?$formel$?}\qquad\=\kill
{\verb?%?}
  \> leitet Kommentare ein\\
{\verb?~?}
  \> festes Leerzeichen (z.\,B.\ in {\verb?5~cm?})\\
{\verb?{..}?}
  \> klammert Textbereiche ein (z.\,B.\ für eine besondere
     Formatierung)\\
{\verb?$formel$?}
  \> klammert Formeln im Fließtext ein\\
{\verb?_?}
  \> tiefstellen (nur im mathematischen Modus)\\
{\verb?^?}
  \> hochstellen (nur im mathematischen Modus)
\end{tabbing}
```

Wenn Ihnen diese einfache Form von Tabellen nicht ausreicht, bietet LaTeX die `tabular`-Umgebung an: Damit können Sie Tabellen mit wechselnder Spaltenbreite, mit Umrandung etc. erzeugen. Wenn Sie außerdem noch die `table`-Umgebung einsetzen, wird die Tabelle je nach Platzangebot automatisch wie ein Bild platziert, ohne dass Löcher im Text entstehen.

3.10 Tabellen

Die wichtigsten Möglichkeiten zum Setzen von Tabellen in LaTeX bieten die Umgebungen `tabular`, `tabular*` und `array`. Diese drei Umgebungen sind bereits in den Standarddokumentenklassen enthalten, wobei die `array`-Umgebung nur im mathematischen Modus verwendet werden kann. Die Syntax und die Bedeutung der Parameter aller drei Umgebungen stimmen überein.

Tabellen können beliebig ineinander verschachtelt werden, man kann sogar innerhalb einer Zelle einer Tabelle wieder eine Tabelle einbauen, es sollte jedoch jede Einzeltabelle innerhalb einer Gruppe eingeschlossen sein, was bei `tabularx`-Tabellen sogar zwingend erforderlich ist.

Tabellen werden von LaTeX als untrennbare Einheit betrachtet, sie müssen deshalb immer auf die Seite passen. Passt eine Tabelle nicht mehr auf eine Seite, wird sie auf die folgende Seite gesetzt. Es gibt aber Pakete, die mehrseitige Tabellen ermöglichen.

Tabellen bestehen aus der Tabellenpräambel, in welcher der Typ der Tabelle und die Formatierung der Spalten deklariert werden, und dem Tabellenkörper, der von den Einträgen gebildet wird. Für die `tabular`-Umgebungen ergibt sich somit folgender Aufbau:

```
% Tabellenpräambel:
\begin{tabular}[position]{spaltenformat}
% Tabellenkörper:
Einträge
\end{tabular}
```

Alle Angaben in der Tabellenpräambel gelten global für die gesamte Tabelle, es gibt jedoch Befehle, mit denen man im Tabellenkörper die globale Festlegung für einzelne Zellen oder Zeilen ändern kann. Die drei Umgebungen haben unterschiedliche Eigenschaften:

- `tabular` für „normale“ Tabellen. Die Breite hängt von den Einträgen und den Abständen zwischen den Spalten ab, LaTeX richtet die Breite der einzelnen Spalten nach den Zelleneinträgen aus. Man muss nur darauf achten, dass die Tabelle nicht zu breit wird.
- `tabular*` hat einen zusätzlichen Parameter, der explizit die Gesamtbreite der Tabelle vorgibt: (`\begin{tabular*}{breite}[position]{spaltenformat}`
- `array` dient der Darstellung von Matrizen, Tensoren usw. und kann nur im mathematischen Modus verwendet werden.

Die Ausrichtung einer Tabelle erfolgt auf die laufende Zeile des Textes und wird immer durch den Positions-Parameter festgelegt. Diesr Parameter ist optional; wird er nicht angegeben, erfolgt eine zentrierte Ausrichtung.

t Die Tabelle wird so in den Text eingefügt, dass die oberste Tabellenzeile bündig mit dem laufenden Text abschließt.

c Die Tabelle wird so in den Text eingefügt, dass sie zentriert zum laufenden Text steht.

b Die Tabelle wird so in den Text eingefügt, dass die unterste Tabellenzeile bündig mit dem laufenden Text abschließt.

Wird die Tabelle als Gleitobjekt (Umgebung `table`, siehe unten) gesetzt, kann der Parameter `position` in der Tabellenpräambel weggelassen werden, da die `table`-Umgebung eine höhere Priorität besitzt und somit die Ausrichtung bestimmt.

Mit dem Parameter `spaltenformat` werden die Anzahl der Spalten, die Ausrichtung der Einträge sowie die Rahmen- und Zwischenlinien der Tabelle definiert. Mit weiteren Befehlen kann gegebenenfalls auch die Breite der Spalten oder ein Text zwischen zwei Spalten festgelegt werden. Die einfachen Formatierungsoptionen sind:

`|`: vertikale Linie über die gesamte Tabellenhöhe
`l`: linksbündige Einträge
`c`: zentrierte Einträge
`r`: rechtsbündige Einträge

Besondere Optionen für das Spaltenformat sind:

- **p**{**breite**}: Eine so genannte `parbox`-Spalte der Breite `breite`, wobei der Eintrag am oberen Rand der Box ausgerichtet wird. Damit lassen sich mehrzeilige Texte in einer Spalte unterbringen (siehe unten).
- *{**anzahl**}{**format**}: Abkürzende Schreibweise für ein Format, dabei treten die in `format` angegebenen Optionen `anzahl`-mal auf
- `@`{**text**}: Diese Option fügt den angegebenen Text oder Befehl an der Stelle in die Tabelle ein, an der die Option in der Präambel steht.

Wird `p{breite}` als Option verwendet, steht der Eintrag innerhalb einer Box, deren Rahmen nicht sichtbar ist. Die Ausrichtung des Textes erfolgt am oberen, nicht sichtbaren Rand der Box. Dieser Befehl wird verwendet, wenn man die Breite der einzelnen Spalten explizit vorgeben will oder ein längerer Text mehrzeilig in der Spalte stehen soll. LaTeX umbricht die Zeilen der Box in Abhängigkeit von der vorgegebenen Breite, es ist aber auch möglich, den Zeilenumbruch mit den Befehlen `\newline` oder `\linebreak` vorzugeben (nicht jedoch mit `\\` oder `\tabularnewline` versuchen, da dies eine neue Zeile in der Tabelle ergibt).

Bei der Verwendung der Option `@{text}` wird der vorgegebene Abstand zwischen den beiden Spalten auf 0 gesetzt und stattdessen der Text eingefügt. Will man den Spaltenabstand unterdrücken, kann man das mittels `@{}` erreichen.

Im Tabellenkörper werden die eigentlichen Daten der Tabelle angegeben und die horizontalen Linien der Tabelle formatiert. Die Eintragung der Daten in die Tabelle erfolgt zeilenweise, einzelne Spalteneinträge werden durch ein & voneinander getrennt. Die Anzahl der Spalten muss mit der in der Tabellenpräambel definierten Spaltenzahl übereinstimmen. Die Tabellenzeilen werden durch den Befehl \\ oder `\tabularnewline` abgeschlossen (Pflicht!).

Nun wird es aber Zeit für ein paar Beispiele:

```
\begin{tabular}{lcr}
{\bf Links} & {\bf Mitte} & {\bf Rechts} \\
111 & 222 & 333 \\
Aus & die & Maus \\
\end{tabular}
```

Ergibt:

Links	**Mitte**	**Rechts**
111	222	333
Aus	die	Maus

Mit senkrechten Strichen sieht das dann folgendermaßen aus:

```
\begin{tabular}{l|c|r}
{\bf Links} & {\bf Mitte} & {\bf Rechts} \\
111 & 222 & 333 \\
Aus & die & Maus \\
\end{tabular}
```

Führt zu:

Links	**Mitte**	**Rechts**
111	222	333
Aus	die	Maus

Hat die Tabelle längere Einträge, gehen bei den Spaltenformaten „r“, „c“ und „l“ die Zeilen irgendwann über den Rand hinaus. Da hilft dann nur noch das Format „p“, wie in folgendem Beispiel (Tabelle 3.1, das auch gleich noch weitere Informationen über die Gestaltung von Tabellen liefert. Hier setzen wir auch die vertikale und horizontale Spaltenbegrenzung ein (im Listing wurde der Text gekürzt):

Tabelle 3.1: Tabelle mit zwei Spalten, die längeren Text enthalten

`\hline`	setzt eine horizontale Linie über die gesamte Breite der Tabelle. Dieser Befehl darf nur am Anfang einer Tabellenzeile verwendet werden.
`\cline{s1-s2}`	setzt ein horizontales Linienstück von Spalte *s1* bis Spalte *s2* der Tabelle. Dieser Befehl darf nur am Anfang einer Tabellenzeile verwendet werden.
`\vline`	setzt eine vertikale Linie über die Höhe der Zeile. `\vline` wird verwendet, wenn nicht die gesamte Tabelle eine vertikale Linie erhalten soll.
`\multicolumn{anz}{form}{text}`	macht aus den nächsten `anz` Spalten eine Spalte von der Gesamtbreite der `anz` Spalten einschließlich Zwischenräumen. Erlaubt sind die Format-Optionen `l`, `c` und `r` sowie ggf. senkrechte Striche davor oder danach, ebenso der `@{text}`-Befehl.

```
\begin{tabular}{|p{0.45 \textwidth}|p{0.45 \textwidth}|}
\hline
\verb?\hline? & setzt eine horizontale Linie über die ... \\
\hline
\verb?\cline{s1-s2}? & setzt ein horizontales Linienstück ... \\
```

```
\hline
\verb?\vline? & setzt eine vertikale Linie über die Höhe ... \\
\hline
\verb?\multicolumn{anz}{form}{text}? & macht aus den nächsten ... \\
\hline
\end{tabular}
```

Der Befehl `cline{s1-s2}` darf auch mehrmals nacheinander auftreten (ebenso `\hline`). Soll die horizontale Linie sich nur über eine Spalte erstrecken, sind `s1` und `s2` gleich – es müssen immer zwei Werte angegeben werden.

Einen anders ausgerichteten Text in einer Zelle gegenüber der restlichen Tabelle erhält man, wenn man den Befehl `\multicolumn{1}{format}{text}` verwendet – also nur für eine Spalte. Da der Befehl die in der Tabellenpräambel vorgegebene Formatierung überdeckt, müssen ggf. senkrechte Striche mit angegeben werden.

```
\begin{tabular}{|l|l||l|}
\hline
{\bf System} & {\bf Multiuser} & {\bf Multitasking} \\
\hline
\hline
MS-DOS & Nein & Nein \\
\cline{1-2}
CP/M & Nein &  \\
\hline
Windows & Nein & Ja \\
\cline{1-2}
Linux & Ja &  \\
\hline
\end{tabular}
```

Ergibt:

System	**Multiuser**	**Multitasking**
MS-DOS	Nein	Nein
CP/M	Nein	
Windows	Nein	Ja
Linux	Ja	

Vorsicht ist bei schmalen Tabellen geboten, da es zu Problemen mit der Worttrennung kommen kann. LaTeX trennt das erste Wort normalerweise nicht. Bei sehr schmalen Spalten kann das aber erforderlich sein (wenn es sich beispielsweise sowieso nur um ein langes Wort handelt). In solchen Fällen hilft das Einfügen eines horizontalen Freiraums der Länge 0 vor dem Wort (`\hspace{0 pt}`). Der wird von LaTeX wie ein Wort betrachtet – und das zweite Wort kann getrennt werden. Dazu ein Beispiel (man beachte das gruselige „fff“):

```
\begin{tabular}{|p{1.5cm}|p{3.0cm}|}
\hline
Donaudampfschifffahrtsgesellschaftskapitän &
Donaudampfschifffahrtsgesellschaftskapitän \\
\hline
\end{tabular}
```

Ergibt:

Donaudampfschifffahrtsgesellschaftskapitän	Donaudampfschifffahrtsgesellschaftskapitän

Dagegen erreicht man mit:

```
\begin{tabular}{|p{1.5cm}|p{3.0cm}|}
\hline
\hspace{0pt}Donaudampfschifffahrtsgesellschaftskapitän &
\hspace{0pt}Donaudampfschifffahrtsgesellschaftskapitän \\
\hline
\end{tabular}
```

Das gewünschte Ergebnis:

Donau- dampf- schiff- fahrts- gesell- schafts- kapitän	Donaudampf- schifffahrtsgesell- schaftskapitän

Innerhalb eines jeden Feldes der Tabelle sind wieder alle Formatierungsangaben sowie Veränderungen der Schriftgröße und -art möglich.

Aber nicht nur die Breite der Spalten kann beeinflusst werden. Mithilfe des Befehls `\vspace{Hoehe}` kann auch die Höhe einer Tabellenzeile angepasst werden und Sie können so die Größe der Tabellenfelder in beiden Dimensionen festlegen, z. B.:

```
\begin{tabular}{|p{1cm}|p{1cm}|p{1cm}|}
\hline
\vspace{1cm} & \vspace{1cm} & \vspace{1cm} \\
\hline
\vspace{1cm} & \vspace{1cm} & \vspace{1cm} \\
\hline
\vspace{1cm} & \vspace{1cm} & \vspace{1cm} \\
\hline
\end{tabular}
```


Schwierig wird es für den Anfänger, wenn Zahlen formatiert werden sollen. Bei der folgenden Tabelle gibt es zwei Spalten. In der ersten Spalte steht eine Mengenangabe, in der zweiten eine Bezeichnung. Die Tabelle ist als solche im Text gar nicht zu erkennen. Im Gegensatz zur `\tabbing`-Umgebung haben Sie aber die Möglichkeit der rechts- bzw. linksbündigen Positionierung:

```
\begin{tabular}{rl}
100 & RAMs 512 MByte, PC 133 \\
52 & RAMs 256 MByte, PC 133 \\
10 & PS2-RAMS 64 MByte \\
350 & Festplatten 80 GByte\\
\end{tabular}
```

Stellt sich im Text dar als:

100	RAMs 512 MByte, PC 133
52	RAMs 256 MByte, PC 133
10	PS2-RAMS 64 MByte
350	Festplatten 80 GByte

Was aber, wenn auch noch Dezimalstellen ins Spiel kommen, etwa bei einer Preisliste. Kein Problem, wenn Sie den Trick kennen. Bei den Preisen werden einfach die Stellen vor dem Komma und jene dahinter in zwei Tabellenspalten untergebracht. Der Dezimalpunkt oder das Dezimalkomma dienen dann als Spaltentrenner:

```
\begin{tabular}{|l|r@{,}l@{ Euro }|}
\hline
RAMs 512 MByte, PC 133 & 45 & 50 \\
\hline
RAMs 256 MByte, PC 133 & 28 & 99 \\
\hline
PS2-RAMS 64 MByte & 9 & 95 \\
\hline
```

```
Festplatten 80 GByte & 121 & 60 \\
\hline
\end{tabular}
```

Ratz-Fatz ist eine ordentliche Preisliste fertig. Mit ein paar Zeilen Perl könnte man den LaTeX-Quelltext sogar aus den Daten einer Kalkulationstabelle oder den Ergebnissen einer Datenbankabfrage vollautomatisch generieren. Für den speziellen Fall der Zahlenformatierung gibt es natürlich ein passendes Makropaket – was dem Beispiel aber keinen Abbruch tut.

RAMs 512 MByte, PC 133	45,50 Euro
RAMs 256 MByte, PC 133	28,99 Euro
PS2-RAMS 64 MByte	9,95 Euro
Festplatten 80 GByte	121,60 Euro

3.11 Gleitobjekte

Leider gibt es bei solchen Tabellen, aber auch bei Bildern, ein kleines Problem. Die Tabelle wird als zusammenhängender Teil des Satzsystems betrachtet – im Grunde wie ein einzelner Buchstabe. Passt sie nicht mehr komplett auf die Seite, wird sie auf die folgende Seite gesetzt.

Beim Gestalten von Artikeln und Büchern gilt die Regel, dass Tabellen, Bilder usw. auch etwas verschoben gesetzt werden dürfen – etwa auf benachbarte Seiten. Der Text läuft dann beispielsweise bis zum Ende der Seite weiter und auf der folgenden Seite erscheint dann die Tabelle. In diesem Fall kann man Tabellen, Bilder usw. auch mit einer Legende (Bildunterschrift) und Nummerierung versehen und sich im Text darauf beziehen („… weitere Optionen sind in Tabelle 3.6 aufgeführt …“).

Solche Textelemente nennt man in LaTeX „Gleitobjekte“, weil sie eben im Text verschieblich (gleitend) angeordnet sind. Manchmal machen solche Gleitobjekte Probleme bei der Anordnung. Kann ein Objekt nicht platziert werden, schiebt das LaTeX-System das Objekt ans Kapitelende (oder bis zum nächsten `\newpage`). Alle folgenden Gleitobjekte werde dahintergepackt, weil ja die Reihenfolge erhalten bleiben muss, und schon stehen alle Tabellen ganz hinten. Da hilft dann nur Trixen, z. B. die Tabelle etwas weiter nach vorne setzen oder in zwei Abteilungen aufteilen. Auch ein `\newpage`-Befehl an der richtigen Stelle hilft. Dass das Problem nicht trivial ist, zeigen etliche Dokumente zur Platzierung von Gleitobjekten in der LaTeX-Dokumentation.

Für Tabellen steht die Gleitobjektumgebung `table` zur Verfügung, deren Grundkonzept kurz vorgestellt werden soll. Das Grundgerüst der `table`-Umgebung ist wie jede andere Umgebung in LaTeX aufgebaut:

```
\begin{table}[ausrichtung]
{Inhalt}
\end{table}
```

Möchte man in einem zweispaltiges Layout eine Tabelle über die gesamte Seitenbreite setzen, so ist die `table*`-Umgebung zu verwenden, die im Übrigen die gleichen Eigenschaften besitzt.

Für die Positionierung der Umgebung dient der Parameter `ausrichtung`, welcher optional verwendet wird. Für diesen Parameter stehen fünf Angaben zur Verfügung:

h LaTeX setzt das Gleitobjekt an der Stelle, wo es im Quelltext vorkommt, wenn dies möglich ist, andernfalls setzt LaTeX das Gleitobjekt an den Seitenanfang. Bei zwei aufeinanderfolgenden „h“-Gleitobjekten ist es besser, diese in eine einzige Gleitobjektumgebung zu setzen.

t Das Gleitobjekt wird am oberen Seitenrand gesetzt.

b Das Gleitobjekt wird am unteren Seitenrand gesetzt.

p Das Gleitobjekt wird auf einer Gleitobjektseite gesetzt, alle folgenden Gleitobjekte können erst nach Ausgabe dieser Seite gesetzt werden.

! hinter h, t oder b bewirkt, dass LaTeX seine Voreinstellungen weitgehend außer Acht lässt, um die vorgeschriebene Positionierung vorzunehmen

Es können mehrere Parameter verwendet werden, wobei die Auswertung nach der angegebenen Reihenfolge erfolgt. Die Verwendung der Parameter ist optional.

Außerdem besteht die Möglichkeit, dass Gleitobjekt, in unserem Fall die Tabelle, durch den Befehl `\caption{text}` mit einem Text zu beschriften und mit einem Label zu versehen (`\label{labeltext}`). Die Tabellen werden dann kapitelweise in der Form „Kapitelnummer.Tabellennummer“ nummeriert. Mit dem Kommando `\ref{labeltext}` können Sie sich im Fließtext dann auf die Nummer der Tabelle beziehen und mit dem Befehl `\listoftables` lässt sich ein Tabellenverzeichnis erstellen.

Die Tabelle kann mit dem Befehl `\caption` in der Gleitobjektumgebung mit einer Legende versehen werden. Die eigentliche Tabelle steht innerhalb der `table`-Umgebung. Insgesamt sieht das dann aus wie im folgenden Listing, wo die Tabelle auch gleich noch zentriert wird:

```
\begin{table}[h!t]
\begin{center}
\caption{Tabellenüberschrift \label{tra-la-la}}
\begin{tabular}
...
\end{tabular}
\end{center}
\end{table}
```

3.12 Aufzählungen

In LaTeX sind mehrere Möglichkeiten zur Formatierung von Aufzählungen vorgesehen. Die einfachste Variante stellt die `itemize`-Umgebung dar. Die `itemize`-Umgebung wird zur einfachen Eingabe von Aufzählungen, Listen oder Ähnlichem benutzt; es ist eine Verschachtelung bis zur Tiefe 4 möglich, mit jeder Verschachtelung wird etwas mehr nach rechts eingerückt. Als Marken werden standardmäßig in der ersten Ebene kleine ausgefüllte Kreise, in der zweiten Ebene Spiegelstriche, in der dritten ein Sternchen und in der vierten Ebene ein Punkt verwendet. Der Text wird wie sonst auch im Blocksatz umbrochen, gleichzeitig werden die Einträge gegenüber dem normalen Text eingerückt.

```
\begin{itemize}
\item Text des ersten Punktes
\item Zweiter Punkt
\end{itemize}
```

Die Marken können auf Wunsch auch geändert werden; soll z. B. in der ersten Ebene ein Spiegelstrich und in der zweiten Ebene ein Plus erscheinen, so kann dies mittels

```
\renewcommand{\labelitemi}{--}
\renewcommand{\labelitemii}{+}
```

geschehen (die dritte und vierte Ebene werden dabei nicht verändert). Mit `\renewcommand` wird ein bestehendes Kommando durch etwas Neues, in diesem Fall durch – und +, ersetzt. Diese Kommandos können auch in der Präambel stehen, wenn sie für das ganze Dokument global gelten sollen. Näheres dazu finden Sie weiter unten im Text. Für alle vier Ebenen gibt es die passende Variable:

`\labelitemi` für die erste Ebene,
`\labelitemii` für die zweite Ebene,
`\labelitemiii` für die dritte Ebene und
`\labelitemiv` für die vierte Ebene.

Wenn statt `itemize` der Umgebungsname `enumerate` verwendet wird, verwendet LaTeX statt der Aufzählungspunkte Zahlen (1., 2. etc.). Beide Umgebungen können ineinander verschachtelt werden. Dabei werden je nach Schachtelungstiefe unterschiedliche Symbole für die Aufzählungspunkte bzw. unterschiedliche Nummerierungsziffern (Kleinbuchstaben, römische Ziffern etc.) verwendet. Außerdem werden die Aufzählungspunkte unterschiedlich stark eingerückt. Auch hier ist eine Verschachtelung bis zur Tiefe 4 möglich. Als Marken werden standardmäßig in der ersten Ebene arabische Ziffern, gefolgt von einem Punkt, in der zweiten Ebene kleine Buchstaben in runden Klammern, in der dritten kleine römische Ziffer, gefolgt von einem Punkt und in der vierten Ebene große Buchstaben, gefolgt von einem Punkt verwendet.

Wie oben beschrieben können auch hier die Labels umdefiniert werden. Es steht

`\labelnumi` für die erste Ebene,
`\labelnumii` für die zweite Ebene,
`\labelnumiii` für die dritte Ebene und
`\labelnumiv` für die vierte Ebene.

Die zugehörigen Zähler heissen `enumi` für den Zähler der ersten Ebene, `enumii` für die zweite Ebene, `enumiii` für die dritte und `enumiv` für die vierte Ebene. Mögliche Darstellungsarten sind:

`\arabic{zähler}` für arabische Ziffern (1, 2, 3, 4, ...),
`\roman{zähler}` für kleine römische Ziffern (i, ii, iii, iv, ...),
`\Roman{zähler}` für große römische Ziffern (I, II, III, IV, ...),
`\alph{zähler}` für kleine Buchstaben (a, b, c, d, ...) und
`\Alph{zähler}` für große Buchstaben (A, B, C, D, ...).

Beispiel:

```
\renewcommand{\labelenumi}{\Alph{enumi}.}
\renewcommand{\labelenumii}{\Roman{enumii}.}
```

Mit der `description`-Umgebung kann eine Aufzählung mit eigenen Marken erreicht werden. Die Aufzählungspunkte werden durch `\item[marke]` festgelegt. Die `marke` kann dabei weggelassen werden. Ansonsten wird sie fett gedruckt. Beispiel:

```
\begin{description}
\item[Erste Marke] Der Text zur ersten Marke
\item[Zweite Marke] Der Text zur zweiten Marke
\item Text zur dritten Marke fehlt
\end{description}
```

Erste Marke Der Text zur ersten Marke

Zweite Marke Der Text zur zweiten Marke

Der Text zur dritten Marke fehlt

3.13 Boxen und Rahmen

Die *Box* spielt bei TEX und LATEX eine entscheidende Rolle. Buchstaben, Zeilen, Seiten – einfach alle Elemente eines Dokuments – sind aus Boxen zusammengesetzt, die normalerweise unsichtbar sind. Intern baut TEX aus Buchstabenboxen Wortboxen auf, aus diesen Zeilenboxen, aus diesen Absatzboxen, aus diesen mit Seitenkopf und -fuß die Seitenboxen. Es gibt aber auch Boxen, die der Benutzer verwenden kann. Sie enthalten meist Text oder andere Satzelemente und – das Wichtigste – sie werden wie ein einzelnes Zeichen behandelt. LATEX stellt dem Anwender drei Arten von Boxen zur Verfügung: *LR-Boxen* für Boxen, in denen der Inhalt horizontal angeordnet ist, *PAR-Boxen* für Boxen, die ganze Absätze enthalten und *RULE-Boxen* für Boxen, die nur Farbe enthalten. Boxen können auch ineinander verschachtelt werden, aber eine Box kann nicht am Zeilen- oder Seitenende umbrochen werden.

3.13.1 LR-Boxen

LR-Boxen	
`\mbox{text}`	für eine einfache LR-Box
`\fbox{text}`	für eine einfache gerahmte LR-Box
`\makebox[breite][pos]{text}`	für eine LR-Box
`\framebox[breite][pos]{text}`	für eine gerahmte LR-Box

Der `\mbox`-Befehl dient nicht nur dazu, ein Wort zusammenzuhalten, sondern auch, um beispielsweise im Mathematik-Modus einen „normalen“ Text in eine Formel einzufügen. Mit dem `\fbox`-Befehl kann man einen Textteil einrahmen, worauf wir weiter unten noch eingehen. Als Position kann „l“ für „linksbündig“ und „r“ für „rechtsbündig“ angegeben werden, die Standardeinstellung ist „zentriert“.

Mit `\raisebox{lift}[oberlänge][unterlänge]{text}` kann eine Box analog zu `\mbox` erzeugt werden, die aber gegenüber der Grundlinie um das Maß *lift* angehoben wird. Ein negatives

Maß senkt die Box ab, z. B. $^{\text{hoch}}$ oder $_{\text{tief}}$. Optional kann man eine Ober- und Unterlänge der entstehenden Box angeben.

3.13.2 PAR-Boxen

PAR-Boxen werden mit dem Kommando `\parbox{breite}[position]{text}` oder analog mit der unten behandelten `minipage`-Umgebung erzeugt. Innerhalb der Box wird der Absatz ganz normal formatiert. Bei der Position kann „t“ oder „b“ angegeben werden, dabei wird bei „t“ die unterste Zeile der Box mit der aktuellen Zeile ausgerichtet, bei „b“ die oberste. Ohne Angabe wird die Box vertikal zentriert zur laufenden Zeile ausgerichtet. Der folgende Absatz zeigt ein Beispiel:

```
\parbox{35mm}{Dies ist eine 30 mm breite Box.
              Sie erscheint vertikal zentriert.}
\hfill die laufende Zeile \hfill
\parbox{55mm}{Dies ist eine 50 mm breite Box.
              Sie erscheint vertikal zentriert.}
```

Dies ist eine 30 mm breite Box. Sie erscheint vertikal zentriert. die laufende Zeile Dies ist eine 50 mm breite Box. Sie erscheint vertikal zentriert.

3.13.3 RULE-Boxen

RULE-Boxen werden mit dem Kommando `\rule[lift]{breite}{höhe}` erzeugt. Der Wert von *lift* gibt an, wie weit diese RULE-Box über die Grundlinie angehoben werden soll; negative Maßangaben senken die RULE-Box gegen die Grundlinie ab. Als Standardwert wird 0 mm angenommen. Auch dazu ein Beispiel:

Hier in dieser Zeile taucht ein schwarzes Rechteck der Breite von 10 mm und der Höhe von 3 mm auf ▬. Eine RULE-Box der Breite 0 mm ist erlaubt! Damit lassen sich Stützen konstrieren, wie z. B. Text ohne Stütze und Text mit Stütze (erzeugt durch `\fbox{\rule[-4mm]{0mm}{10mm}Text}`). Die umgebende Box wird im zweiten Fall vertikal ausgedehnt! Auch andere Tricks lassen sich mit RULE-Boxen vollführen, etwa ein kleines graues Kästchen als erstes Aufzählungszeichen:

```
\renewcommand\labelitemi {\textcolor[gray]{0.5}{\rule{1.0ex}{1.0ex}}}
```

3.13.4 Rahmen

Zur Hervorhebung von Texten können diese in einen Rahmen gestellt werden. Das dazu erforderliche Kommando lautet `\fbox{text}` und liefert Resultate wie diesen Text. Wenn mehrzeilige Blöcke eingerahmt werden sollen, muss mit `\fbox` eine `minipage`-Umgebung definiert werden.

Im folgenden Beispiel wird ein kleiner Textblock zentriert und mit `\fbox` eingerahmt. Beim Einsatz von `\hfill` muss beachtet werden, dass damit nur ein Füllabstand zwischen vorhandenen Objekten eingefügt werden kann. Daher muss vor und nach `\hfill` mit `\hbox{}` eine leere Textbox angegeben werden.

```
\hbox{}          % leere Box links
\hfill           % Füllabstand zur minipage
\fbox{
\begin{minipage}{5cm}
Hier wurde eine 5 cm breite Minipage durch ein vor-
und ein nachgestelltes {\tt\textbackslash{}hfill}-Kom\-man\-do
zentriert und mit {\tt\textbackslash{}fbox} eingerahmt.
\end{minipage}}
\hfill           % Füllabstand zur nächsten Box
\hbox{}          % leere Box
```

Hier wurde eine 5 cm breite Minipage durch ein vor- und ein nachgestelltes `\hfill`-Kommando zentriert und mit `\fbox` eingerahmt.

Innerhalb von `\fbox` darf das Kommando `\verb_text_` nicht verwendet werden. Diese Einschränkung können Sie mit der Umgebung `lrbox` umgehen.

3.13.5 Box-Register

Mit `\newsavebox{\name}` wird eine Variable für eine Box mit dem angegebenen Namen angelegt. Diese können Sie mittels `\sbox{\name}{text}` oder `\savebox{\name}[breite][position]{text}` mit einem Inhalt versehen. Der Unterschied zwischen diesen beiden Kommandos ist wie zwischen `\mbox` und `\makebox`. Der Inhalt der Box wird mit dem Befehl `\usebox{\name}` ausgegeben:

```
\newsavebox{\ownbox}
\sbox{\ownbox}{ganz toller Text}
Hier kommt ein \usebox{\ownbox}.
\savebox{\ownbox}[3cm][l]{Text}
Hier kommt ein \usebox{\ownbox}.
```

Hier kommt ein ganz toller Text. Hier kommt ein Text .

3.13.6 Mehrspaltiger Text

Bereits auf Seite 22 wurde beschrieben, dass durch die Option `twocolumn` im Kommando `\documentclass` der gesamte Text zweispaltig angeordnet werden kann. Für viele Problemstellungen ist diese Lösung aber ungeeignet. Oft sollen nur kleine Textportionen nebeneinander angeordnet werden, während der restliche Text einspaltig über die volle Breite geht. Auch für solche Situationen bietet LaTeX mehrere Varianten an, von denen hier wiederum nur die wichtigste vorgestellt wird: der Umgang mit der `minipage`-Umgebung. Die Syntax sieht folgendermaßen aus:

minipage-Umgebung

```
\begin{minipage}[t]{4cm}  % die erste, 4 cm breite Minipage
text ...                  % der Text in der ersten Minipage
\end{minipage}
\hfill                    % Abstand zwischen den beiden Minipages
\begin{minipage}[t]{4cm}  % die zweite Minipage
text ...
\end{minipage}
\hfill
\begin{minipage}[t]{4cm}  % die dritte Minipage
text ...
\end{minipage}
```

Die Breite aller angegebenen `minipage`-Umgebungen darf die gesamte Textbreite nicht überschreiten. Der optionale Parameter `t` (top) bewirkt, dass die Textblöcke vertikal an der oberen Kante ausgerichtet werden. Alternativ könnten Sie `b` (bottom) angeben, um die Textblöcke an der unteren Kante auszurichten, oder ganz auf den Parameter verzichten, um die Blöcke vertikal zu zentrieren. Das Kommando `\hfill` zwischen den Blöcken bewirkt, dass der verbleibende horizontale Freiraum zwischen ihnen verteilt wird.

```
\begin{minipage}[t]{5.5cm}
Das ist der erste 5,5 cm breite Textblock.
\end{minipage}
\hspace{0.5cm}

\begin{minipage}[t]{5.5cm}
Der zweite Block wird daneben platziert und ist ebenso breit. Die
beiden Blöcke werden an ihrer oberen Kante ausgerichtet.
\end{minipage}
```

Das Ergebnis haben wir zur Verdeutlichung mit einer Box umgeben:

Das ist der erste 5,5 cm breite Textblock.	Der zweite Block wird daneben platziert und ist ebenso breit. Die beiden Blöcke werden an ihrer oberen Kante ausgerichtet.

3.14 Explizite Positionierung

Gleitobjekte, Minipages etc. erlauben schon sehr viele Freiheiten, die man aber nicht immer und unbedingt nutzen sollte. Für manche Aufgaben wird jedoch eine explizite Positionierung von Objekten nötig – oder man quält sich unnötig.

Für das Positionieren von Elementen in einem Gitter gibt es für LaTeX zahlreiche Pakete, z. B. grid, grid-system, gridset, lpic, ltxgrid, pas-tableur, textpos, typogrid, vgrid etc. Viele dieser Pakete dienen sehr speziellen Anwendungen und sind und nicht allgemein nutzbar. Oft sind sie auch nicht in Standarddistributionen enthalten und müssen von *www.ctan.org* heruntergeladen und dann installiert werden. Der Weg zu diesen Paketen kann insofern etwas steinig sein, weil Sie gegebenenfalls noch etliche unterstützende Pakete herunterladen und installieren müssen.

Es sollen daher hier nur zwei Pakete vorgestellt werden, die eine einfache explizite Positionierung erlauben. Dies kann entweder durch Setzen eines beliebigen Objekts an eine bestimmte Position (X,Y) auf der Seite geschehen oder auch durch relatives Positionieren zu anderen Objekten auf der Seite.

3.14.1 grid-system

`grid-system` ist ein relativ junges Paket, das sich am Layoutsystem von Cascading Style Sheets (CSS) orientiert. Das Paket ermöglicht die Darstellung von mehrspaltigen Textblöcken, die zueinander ausgerichtet sind.

Das Paket stellt zwei neue Umgebungen, `row` und `cell` zur Verfügung. `row` besitzt zwei Argumente: die Anzahl der Spalten, die der gesamte Absatz haben soll und die Anzahl der dargestellten Zellen. Letztere ist notwendig, um vorab die Breite jeder einzelnen Zelle bestimmen zu können.

Die Umgebung `cell` stellt dann eine Zelle dar, die sich auch über mehrere Spalten erstrecken kann (erstes Argument ist die Spaltenzahl). Die Summe der Spaltenangaben von `cell` muss dabei immer gleich der Anzahl cde mit `row` definierten Spalten sein.

Mit `grid-system` kann man zwar keine Objekte absolut positionieren, aber zur Gestaltung von Postern eignet sich das Paket gut; die zu platzierenden Elemente sollten nur alle gleich hoch sein. Bei älteren Versionen muss noch `\parindent` den Wert 0 haben oder vor die erste Zelle ein `\noindent` gesetzt werden. Andernfalls ragen die definierten Spalten in den rechten Rand hinein. Dazu ein Minimalbeispiel:

```
\begin{row}{3}{2}
  \noindent%
  \begin{cell}{1}
   Hallo. Ich bin ein kleiner einspaltiger Blindtext.
Und zwar schon so lange ich denken kann. Es war nicht
leicht zu verstehen, was es bedeutet, ein blinder Text
zu sein: Man macht keinen Sinn.
\end{cell}
  \begin{cell}{2}
  Dieser zweispaltige Textblock wird zusammenhangslos eingeschoben
und rumgedreht -- und oftmals gar nicht erst gelesen. Er dient
lediglich als Platzhalter, um mal zu zeigen, wie diese Stelle
der Seite aussieht, wenn ein paar Zeilen vorhanden sind.
  \end{cell}
\end{row}
```

`grid-system` stützt sich auch etliche andere Pakete, die eventuell auch installiert werden müssen.

3.14.2 textpos

`textpos` kann einzelne Textblöcke relativ zur aktuellen Position oder Absolut auf der Seite platzieren. Die Option `absolute` bestimmt dieses Verhalten. Die `textblock`-Umgebung orientiert sich an einem (virtuellen) Gitter, das mittels `TPgrid` definiert werden kann. So lassen sich die Textblöcke hirizontal und vertikal „einrasten“.

Die `textblock*`-Umgebung stellt dagegen einen Textblock an einer absoluten Position auf der Seite dar, wobei ab der oberen linken Ecke gemessen wird. Als Argumente werden die Breite des Textblocks

und dessen Position (X, Y) angegeben. Die Höhe der Box wird durch den Inhalt bestimmt, der per Standard im Blocksatz gesetzt wird. Der Inhalt der Box ist beliebig und kann auch Bilder oder andere Gleitobjekte enthalten.

Gibt man die Paket-Option `showboxes` an, wird die jeweilige Box noch umrahmt, was beim Entwurf recht hilfreich ist. Ist die Box recht klein, kann es kleine Probleme beim Umbrechen des Textes geben, da dieser über den Rand hinausgeht. Dies ist bei Blocksatz und solch kleinen Boxen normal, da der maximale Abstand für Wörter sonst nicht eingehalten werden könnte. Vergrößert man die Box oder nutzt `\raggedright`, `\raggedleft` oder `\centering` verschwinden diese Umbruch-Probleme. Das folgende Beispiel zeigt die Anwendung:

```
\documentclass{article}
\usepackage[absolute,showboxes]{textpos}

\begin{document}
Das ist der normale Text der Seite,
der normalerweise nicht benutzt wird.

\begin{textblock*}{40mm}(50mm,80mm)
Hallo. Ich bin ein kleiner Blindtext.
Und zwar schon so lange ich denken kann. Es war nicht
leicht zu verstehen, was es bedeutet, ein blinder Text
zu sein: Man macht keinen Sinn.
\end{textblock*}
\end{document}
```

Ein Ausschnitt der Seite zeigt das Bild 3.1. Hier sieht man auch drei Zeilen, die nicht ganz in die Box passen. Man könnte auch noch den Ursprung in der linken oberen Ecke über `\textblockorigin` verschieben, wenn man beispielsweise einen Rand mit beachten möchte. Diese und weitere Funktionen kann man in der Dokumentation nachlesen.

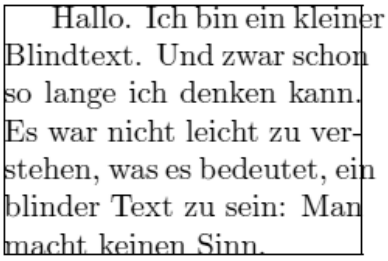

Bild 3.1: Das Ergebnis der Anwendung von `textpos` (Ausschnitt)

4

Abbildungen

Um eine Abbildung in ein LaTeX-Dokument einzubinden, werden zumeist eine `figure`-Umgebung und ein `\includegraphics`-Kommando kombiniert. Die `figure`-Umgebung ist für die Platzierung und Beschriftung der Abbildung verantwortlich, während das Kommando `\includegraphics` aus dem `graphicx`-Paket eine PostScript-Grafikdatei einliest.

4.1 Grafik als Gleitobjekt einbinden

Die `figure`-Umgebung ist auch wieder ein Gleitobjekt (wie bei den Tabellen, Seite 33 beschrieben). Der optionale Parameter dieser Umgebung bestimmt, wie die Grafik im Text platziert wird. Dabei gilt auch für die `figure`-Umgebung das dort gesagte: Als Parameter ist entweder `h` (*here*), `t` (*top*), `b` (*bottom*) oder `p` (*page*) erlaubt – oder eine beliebige Kombination der Buchstaben. Wenn LaTeX dennoch die Abbildung woanders platziert, gibt ein zusätzliches Ausrufezeichen Ihrem Platzierungswunsch mehr Nachdruck (also z. B. `[h!]`).

Platzierung der Abbildung

`h:`	Die Abbildung wird genau an dieser Stelle im Text angezeigt.
`t:`	Die Abbildung wird am Beginn der laufenden Seite platziert.
`b:`	Die Abbildung wird am Ende der laufenden Seite platziert.
`p:`	Mehrere Abbildungen werden auf einer eigenen Seite zusammengefasst.

In zweispaltigen Texten kann statt `figure` der Umgebungsname `figure*` verwendet werden. In diesem Fall erstreckt sich die Abbildung über beide Spalten (anstatt per Default nur die Breite einer Spalte zu nutzen). `figure*` kann nicht mit der Option `h` kombiniert werden.

Zur Beschriftung der Grafik wird `\caption` eingesetzt. LaTeX stellt dem eigentlichen Beschriftungstext „Abbildung *n*:“ voran, wobei für *n* entweder eine laufende Nummer (Texttyp *article*) oder eine Kapitelnummer (3.5 für das fünfte Bild im dritten Kapitel) eingesetzt wird. Wenn innerhalb von `\caption` eine `\label`-Anweisung verwendet wird, kann mit `\ref` auf die Abbildungsnummer und mit `\pageref` auf die dazugehörige Seitennummer verwiesen werden.

figure-Umgebung

```
\begin{figure}[h]

% Kommandos zur Erzeugung der eigentlichen Grafik z. B.
% \includegraphics{...} oder den internen Grafikmöglichkeiten

\caption{\label{marke-fuer-querverweis}Beschriftungstext}
\end{figure}
```

Der Beschriftungstext wird automatisch zentriert. Wenn das Bild schmaler als der Beschriftungstext ist, sollte dieser mit `\parbox{7cm}{\caption{...}}` auf die Breite des Bildes (hier 7 cm) beschränkt werden.

Das graphicx-Paket: Das *graphicx*-Paket stellt unter anderem das Kommando `\includegraphics` zur Verfügung. Damit kann eine Grafikdatei in das LaTeX-Dokument eingebunden werden. Die Syntax für `\includegraphics` sieht so aus:

```
\usepackage{graphicx}
...
\includegraphics[option1=... option2=...]{filename}
```

Die Optionen steuern unter anderem die Größe, Skalierung und Rotierung der Grafik. Die folgende Tabelle fasst die wichtigsten Optionen zusammen.

includegraphics-Optionen	
`width=`*n*	bestimmt die Breite der Grafik im Dokument
`height=`*n*	bestimmt die Höhe der Grafik
`scale=`*n*	gibt an, wie stark die Grafik skaliert werden soll (relativ zur Originalgröße der Grafik)
`angle=`*n*	gibt an, um wie viel Grad die Grafik gedreht werden soll

Zur Größenangabe ist normalerweise die Angabe einer Option ausreichend (`width` oder `height`) – die Grafik wird dann automatisch korrekt skaliert. Bei `width` kann man auch wieder auf die aktuelle Textbreite zurückgreifen, z. B. `width={\textwidth}` oder `width={0.95 \textwidth}`.

Wenn Sie das LaTeX-Dokument später in das PostScript-Format umwandeln (`dvips`) und ausdrucken möchten, kommen für `\includegraphics` nur PostScript-Dateien (EPS, PS) in Frage. Grafiken in anderen Formaten (GIF, TIFF, JPEG, PNG) müssen Sie gegebenenfalls vorher mit einem Bildverarbeitungsprogramm in das EPS-Format umwandeln. Der direkte Import von Bitmap-Grafiken ist nur dann zulässig, wenn Sie Ihr LaTeX-Dokument mit `pdflatex` direkt in eine PDF-Datei umwandeln möchten (siehe Seite 19).

Abbildungen werden generell weder zentriert noch eingerahmt. Wenn diese Formatierungsmerkmale erwünscht sind, können Sie das Bild mit `\begin{center}` ... `\end{center}` oder mit `\centerline` zentrieren und mit `\fbox` in einen Rahmen setzen.

```
\begin{figure}
\begin{center}
\includegraphics[width=5cm]{figure.ps}
\caption{zentriert}
\end{center}
\end{figure}

\begin{figure}
\begin{center}
\fbox{\includegraphics[width=5cm]{figure.ps}}
\caption{zusätzlich gerahmt}
\end{center}
\end{figure}
```

Wenn mehrere Abbildungen nebeneinander platziert werden sollen, müssen wie im folgenden Beispiel innerhalb der `figure`-Umgebung mehrere Minipages platziert werden. Der umgekehrte Fall – also die Verwendung einer `figure`-Umgebung innerhalb einer Minipage – ist nicht möglich. Aus diesem Grund ist es nicht ohne weiteres möglich, eine beschriftete Grafik und Text nebeneinander zu platzieren. Sie können aber eine unbeschriftete PostScript-Grafik durch die direkte Verwendung des `\includegraphics`-Kommandos ohne `figure`-Umgebung in einer Minipage ausgeben.

```
\begin{figure}[ht]
\begin{minipage}[t]{5.5cm}
\begin{center}
\includegraphics[width=5cm]{bilder/b-latex-screenshot.eps}
\caption{\label{latex-fig1} \textsl{Dieser Screenshot ...
  eingebunden.}
\end{center}
\end{minipage}
\hfill
\begin{minipage}[t]{5.5cm}
\begin{center}
```

```
\setlength{\fboxsep}{0mm} %kein Abstand zur Umrahmung
\fbox{\includegraphics[width=5cm]{bilder/b-latex-maple.eps}}
\caption{\label{latex-fig2} \textsl{Dieses Diagramm wurde ...
    eingebunden.}}
\end{center}
\end{minipage}
\end{figure}
```

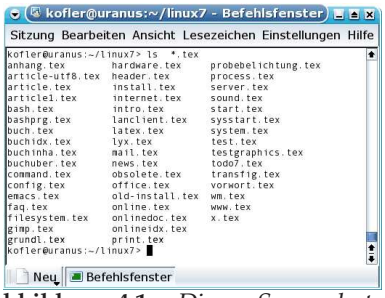

Abbildung 4.1: *Dieser Screenshot ... eingebunden.*

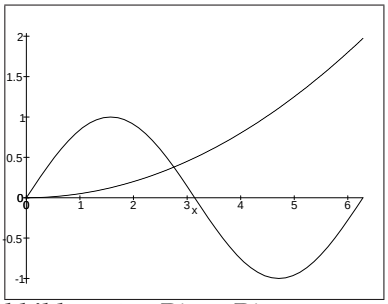

Abbildung 4.2: *Dieses Diagramm wurde ... eingebunden.*

Die Beispielabbildungen 4.1 und 4.2 wurden mit den obigen Kommandos erzeugt. Die Verweise auf die Bildnummern wurden mit `\ref{latex-fig1}` bzw. mit `\ref{latex-fig2}` erzeugt.

Hinweis

Wenn eine PostScript-Grafik nicht an der durch die `figure`-Umgebung vorgesehenen Position erscheint, sondern irgendwo anders auf der Seite, ist meist eine falsche BoundingBox-Information innerhalb der importierten PostScript-Grafik schuld. Die BoundingBox-Zeile gibt die Position und die Größe der Grafik an.

Frühere Versionen von `ksnapshot` produzierten beispielsweise falsche BoundingBoxes, wenn der Screenshot als EPS-Datei gespeichert wird. Speichern Sie den Screenshot stattdessen als Bitmap und verwenden Sie ein anderes Programm, um daraus eine korrekte EPS-Datei zu erzeugen.

Eine Menge weiterer Informationen zum Thema Grafik und Farben finden Sie auf der folgenden, exzellenten Seite im Internet:
http://www.linmpi.mpg.de/ daly/latex/grf.htm

4.2 Grafiken umwandeln

Bei Photos etc. müssen Sie sich eventuell überlegen, ob Sie das Bild gleich als Graustufenbild in Ihr LaTeX-Dokument einbinden wollen (wenn sowieso nur schwarzweiss gedruckt werden soll). Phostos liegen immer als Bitmap vor, bei anderen Bildern (Grafiken, Diagramme usw.) sollten Sie dagegen versuchen, Bitmapgrafiken zu vermeiden. Diese haben den Nachteil, dass sie sich nur schlecht skalieren lassen. Viele Anwendungen verarbeiten ihre Grafiken aber auch vektororientiert und erzeugen deshalb mit Hilfe eines Postscript-Druckertreibers hervorragende Postscriptdateien. Diese Postscriptdateien wandeln Sie dann in EPS (Encapsulated Post Script) um.

Die generierte EPS-Datei laden Sie in GSview, falls die Bounding Box noch angepasst werden muss. Um in GSview die Bounding Box anzuzeigen, wählen Sie „Menü Options“ → „Show Bounding Box“. Sollte die Bounding Box nicht Ihren Wünschen entsprechen, können Sie mit dem Befehl „File“ → „PS to EPS“ die Bounding Box automatisch oder manuell festlegen und die Ausgabe in einer neuen Datei speichern.

Dabei sollten Sie unbedingt beachten, dass für Postscript- oder DVI-Ausgabe, die Bilder im EPS-Format vorliegen müssen, wohingegen beim Verwenden von `pdflatex` nur Grafiken verarbeitet werden, die als PDF, JPG oder PNG vorliegen. Soll also beides, DVI und PDF, erzeugt werden, ist das EPS-Format günstiger, weil Sie über die drei Schritte DVI → PS → PDF beide Formate erzeugen können.

Wenn Sie dagegen nur mit `pdflatex` arbeiten (etwa beim Erstellen einer Präsentation), so können Sie die EPS-Dateien nach PDF umwandeln. Dazu gibt es den Befehl `epstopdf`. Für die Umwandlung von JPG oder PNG in EPS kann neben einem Grafikprogramm der Befehl `bmeps` verwendet werden.

Für die Batch-orientierte Konvertierung von Grafiken (Formatumwandlung, Filterung, Beschneidung und vieles mehr) eignet sich übrigens das ImageMagick-Paket vorzüglich (*http://imagemagick.org/*). Für das automatisierte Erstellen von Diagrammen etc. aus Rohdaten eignet sich das Programm „Gnuplot“. Unter der URL *http://gnuplot.sourceforge.net/* finden Sie weitere Informationen und zum Download gehen Sie nach *http://sourceforge.net/projects/gnuplot/files/*.

4.3 Ganzseitige Grafiken

Manchmal ist es notwendig, eine Grafik ganzseitig in ein LaTeX-Dokument zu übernehmen – ohne den obligatorischen weißen Rand um den Textkörper. Dazu verwenden Sie beispielsweise das folgende Makro. Ein Makro deshalb, weil es doch recht viele Befehle sind.[1] Hier nur soviel: Ein Makro fasst eine Reihe von LaTeX-Befehlen unter einem Namen zusammen. Das folgende Makro definiert einen neuen Befehl, `\includegraphicspage`. Es basiert auf dem graphicx-Paket und nutzt dessen includegraphics-Befehl.

```
\newcommand{\includegraphicspage}[1]{%
  \newpage
  \bgroup
  \thispagestyle{empty}%
  \hoffset=-1in
  \voffset=-1in
  \topmargin=0pt
  \headheight=0pt
  \headsep=0pt
  \evensidemargin=0pt
  \oddsidemargin=0pt
  \parskip=0pt
  \parindent=0pt
  \includegraphics[width=\paperwidth]{#1}
  \newpage
  \egroup}
```

Das Makro setzt alle um den Textkörper befindlichen Satzspiegel-Längen auf Null und erweitert dann den Textkörper auf das gesamte Blatt, das die Abmessungen `\paperwidth` und `\paperheight` besitzt. Da TeX eine Seite per Default immer einen Zoll von oben und einen Zoll von links beginnt, müssen die Werte von `\hoffset` und `\voffset` nicht auf Null sondern auf den den entsprechenden negativen Wert gesetzt werden. Die Nutzung des neuen Befehls innerhalb eines LaTeX-Dokuments ist dann ganz einfach, als Parameter wird der Name der Grafikdatei angegeben:

```
  ...
\includegraphicspage{diagramm.eps}
  ...
```

Ein Beispiel finden Sie auf der letzten Seite (vor dem Stichwortverzeichnis). Die Grafik wird automatisch auf die Größe von `\paperwidth` skaliert, wobei das Verhältnis von Seite zu Höhe des Bildes dem DIN-Format entsprechen muss (sonst wird ggf. in der Höhe etwas abgeschnitten oder es bleibt unten ein weißer Rand. Falls Sie PDF-LaTeXverwenden und die ganzseitige Grafik bereits als PDF-Dokument vorliegt, kann man auch das Paket *pdfpages* verwenden.

[1] Wie Makros erstellt werden, ist in Abschnitt 10 auf Seite 79 genauer erläutert.

5

Mathematische Formeln

Damit in LaTeX mathematische Formeln dargestellt werden, müssen Sie in den Mathematik-Modus umschalten. Sämtliche hier vorgestellten Kommandos können nur in diesem Modus ausgeführt werden!

Zur Aktivierung des Mathe-Modus haben Sie drei Möglichkeiten:

- Die Formel wird im laufenden Text zwischen zwei `$`-Zeichen gesetzt. In diesem Fall versucht LaTeX, die Formel an die Zeilenhöhe anzupassen.
- Die Formel wird in einem eigenen Absatz mit `\[` eingeleitet und mit `\]` beendet. Sie steht dann als eigener Absatz zentriert im Text.
- Die Formel wird in einer `equation`-Umgebung geschrieben. Auch hier wird die Formel in einem eigenen Absatz zentriert dargestellt, wobei sie in der `equation`-Umgebung zusätzlich automatisch nummeriert wird.

LaTeX-Formelmodi

`$formel$`	Formel im Fließtext
`\[ formel \]`	eigenständige Formel

`\begin{equation}`
`  formel`
`\end{equation}`
	eigenständige Formel mit Nummerierung

Beachten Sie bitte, dass Formeln im Fließtext anders aussehen als eigenständige Formeln: Bei `$`-Formeln verwendet LaTeX eine etwas kleinere Schriftart und versucht, die Formel so zu setzen, dass sie vertikal möglichst wenig Platz beansprucht. Insbesondere werden Hoch- und Tiefstellungen bei manchen mathematischen Symbolen (Limes, Integrale, Summen) anders platziert.

Im Folgenden sehen Sie dreimal dieselbe Formel `\int_{i=1}^{n} x^i dx`: Im Fließtext sieht die mit `$` geklammerte Formel so aus: $\int_{i=1}^{n} x^i dx$. Mit `\[ ... \]` bzw. zwischen `\begin{equation}` und `\end{equation}` erhalten Sie dagegen folgende Ergebnisse:

$$\int_{i=1}^{n} x^i dx$$

$$\int_{i=1}^{n} x^i dx \tag{5.1}$$

Wenn Sie möchten, dass eigenständige Formeln nicht zentriert, sondern linksbündig dargestellt werden, können Sie am Beginn des LaTeX-Dokuments die Anweisung `\usepackage{fleqn}` verwenden. Anschließend können Sie mit `\mathindent2cm` eine Einrücktiefe vom linken Rand (hier: 2 cm)

angeben. Alternativ dazu können Sie mit `\usepackage{leqno}` erreichen, dass LaTeX Formeln in der `equation`-Umgebung nicht rechts-, sondern linksbündig nummeriert.

Innerhalb von Formeln wird Text generell kursiv dargestellt. Für Variablen ist das die übliche mathematische Schreibweise. Funktionsnamen sollen dagegen zumeist in aufrechter Schrift dargestellt werden. Dazu existieren eigene LaTeX-Kommandos wie `\sin`. $\sin(x^2)$ wird durch den Text `$\sin(x^2)$` erzeugt. LaTeX kennt die folgenden Kommandos zur Ausgabe von Funktionen:

Schlüsselwörter für mathematische Funktionen

```
\arccos, \arcsin, \arctan, \arg, \cos, \cosh, \cot, \coth, \csc,
\deg, \det, \dim, \exp, \gcd, \hom, \inf, \ker, \lg, \lim, \liminf,
\limsup, \ln, \log, \max, \min, \Pr, \sec, \sin, \sinh, \sup, \tan,
\tanh
```

Der Mathematik-Modus beseitigt auch alle Leerzeichen in der Formel und bestimmt die Abstände zwischen den Elementen nach anderen Regeln. Normaler Text wird daher entstellt. Soll er (oder ein oben nicht angeführter Funktionsname) innnerhalb von Formeln dargestellt werden, muss der Text in einer (unsichtbaren) Box stehen, was Sie mit der Anweisung `\mbox{...}` erreichen. Es sind auch Anweisungen wie `{\it varname}` oder `{\bf X}` erlaubt, um längere Variablennamen auszugeben oder Texte hervorzuheben.

LaTeX geht im Mathe-Modus manchmal recht sparsam mit Abständen um. In der Formel $\sin(xy)$ (`\sin(x y)`) ist kaum zu erkennen, dass x und y zwei eigenständige Variablen sind, die miteinander multipliziert werden. Zur Vergrößerung der Abstände können Sie die drei Kommandos `\,` (kleiner Abstand), `\:` (mittlerer Abstand) und `\;` (großer Abstand) einsetzen: `\sin(x \: y)` wird dann zu $\sin(x\,y)$.

Konstruktion mathematischer Formeln

`a^{b}`	a^b	`\sum_{a}^{b} c`	$\sum_a^b c$
`a_{b}`	a_b	`\prod_{a}^{b} c`	$\prod_a^b c$
`\frac{a}{b}`	$\frac{a}{b}$	`{a \choose b}`	$\binom{a}{b}$
`\sqrt{a}`	$\sqrt{a}$	`\overline{abc}`	$\overline{abc}$
`\sqrt[n]{a}`	$\sqrt[n]{a}$	`\underline{abc}`	$\underline{abc}$
`\int_{a}^{b} c`	$\int_a^b c$	`\overbrace{abc}^{d}`	$\overbrace{abc}^{d}$
`\oint_{a}^{b} c`	$\oint_a^b c$	`\underbrace{abc}_{d}`	$\underbrace{abc}_{d}$

Die oben genannten Kommandos werden zur Zusammensetzung von Brüchen, Wurzeln, Integralen, Summen etc. eingesetzt. Die Kommandos können beliebig verschachtelt werden, um Wurzeln in Brüchen oder Ähnliches zu erreichen. LaTeX kümmert sich um eine geeignete Schriftgröße und andere Formatierungsdetails. Beachten Sie, dass Sie beim Hoch- und Tiefstellen den jeweiligen Ausdruck in geschweifte Klammern setzen – andernfalls gilt ^ bzw. _ nur für das erste nachfolgende Zeichen. Es lassen sich auch recht einfach Formeln aus der Digitaltechnik setzen, beispielsweise ergibt

```
\[ y = (\overline{x1} * \overline{x2} * x3) +
       (\overline{x1} * x2 * \overline{x3)} +
       (x1 * \overline{x2} * \overline{x3}) +
       (x1 * \overline{x2} * x3) + (x1 * x2 * \overline{x3}) \]
```

die Logikgleichung:

$$y = (\overline{x1} * \overline{x2} * x3) + (\overline{x1} * x2 * \overline{x3}) + (x1 * \overline{x2} * \overline{x3}) + (x1 * \overline{x2} * x3) + (x1 * x2 * \overline{x3}).$$

Wenn Sie mit den von LaTeX automatisch gewählten Schriftgrößen nicht zufrieden sind, können Sie diese durch vier Kommandos beeinflussen:

Schriftgröße in mathematischen Formeln	
`\displaystyle`	normal
`\textstyle`	etwas kleiner (Schriftart in $-Formeln)
`\itriptstyle`	noch kleiner (Indizes und Hochzahlen erster Ordnung)
`\itriptscriptstyle`	winzig (Indizes und Hochzahlen zweiter Ordnung)

Die folgenden Formeln demonstrieren das Zusammenspiel von einigen der hier aufgezählten Kommandos. In den Formeln kommen auch einige Kommandos vor, die erst in späteren Abschnitten behandelt werden.

```
\[ \frac{ \frac{a+1}{b-1} }{ \frac{c+1}{d-1}}
   \qquad \mbox{und} \qquad
   \frac{\displaystyle \frac{a+1}{b-1}}
        {\displaystyle \frac{c+1}{d-1}}  \]
```

$$\frac{\frac{a+1}{b-1}}{\frac{c+1}{d-1}} \qquad \text{und} \qquad \frac{\displaystyle \frac{a+1}{b-1}}{\displaystyle \frac{c+1}{d-1}}$$

```
\[ \oint_C f(z) \, dz = \int_0^{2 \pi} f(z(t)) \,
   \frac{d \, z(t)}{dt} \, dt
   \qquad\mbox{mit}\qquad
   z(t)= z0 + r\, (\cos(t) + I\, sin(t)) \]
```

$$\oint_C f(z)\,dz = \int_0^{2\pi} f(z(t))\,\frac{d\,z(t)}{dt}\,dt \qquad \text{mit} \qquad z(t) = z0 + r\,(\cos(t) + I\,sin(t))$$

```
\[   \left[      {\begin{array}{c}
  {\displaystyle \frac {{\frac {{ \partial}}{{ \partial}{u}}}\,
    {\rm f}(\,{u}, {v}, {z}\,)}{{a}\,\sqrt {{\rm sin}(\,{v}\,)^{2} +
    {\rm sinh}(\,{u}\,)^{2}}}} \\[5mm]
  {\displaystyle \frac {{\frac {{ \partial}}{{ \partial}{v}}}\,
    {\rm f}(\,{u}, {v}, {z}\,)}{{a}\,\sqrt {{\rm sin}(\,{v}\,)^{2} +
    {\rm sinh}(\,{u}\,)^{2}}}} \\[5mm]
  {\frac {{ \partial}}{{ \partial}{z}}}\,{\rm f}(\,{u}, {v}, {z}\,)
\end{array}}    \right]   \]
```

$$\left[\begin{array}{c}
\frac{\frac{\partial}{\partial u}\,\mathrm{f}(\,u,v,z\,)}{a\,\sqrt{\sin(\,v\,)^2+\sinh(\,u\,)^2}} \\[5mm]
\frac{\frac{\partial}{\partial v}\,\mathrm{f}(\,u,v,z\,)}{a\,\sqrt{\sin(\,v\,)^2+\sinh(\,u\,)^2}} \\[5mm]
\frac{\partial}{\partial z}\,\mathrm{f}(\,u,v,z\,)
\end{array}\right]$$

5.1 Klammern

Klammern werden prinzipiell direkt mit `(..)` oder `[...]` gebildet. Da geschweifte Klammern in LaTeX eine besondere Bedeutung haben, muss ihnen ein Backslash vorangestellt werden, also `\{...\}`.

In mathematischen Formeln soll die Größe der Klammern im Normalfall dem geklammerten Ausdruck entsprechen. Standardmäßig ist das nicht der Fall. Das Verhalten kann aber mit den Kommandos `\left` und `\right` realisiert werden, die der linken bzw. der rechten Klammer vorangestellt werden. Das vorangegangene Beispiel zeigt die Anwendung dieser beiden Kommandos. `\left` und `\right` können auch vor einigen weiteren mathematischen Symbolen verwendet werden, etwa vor $\rightarrow$ für Beträge.

Will man nur einseitig eine Klammer setzen (nur rechts oder nur links), gibt es die „unsichtbaren" Klammern `\left.` und `\right.`, die dafür sorgen, dass wieder Klammerpaare entstehen. In der folgenden Formel wurde die linke Klammer unsichtbar gemacht:

$$\left.\begin{array}{c} \frac{\partial}{\partial u}\,\mathrm{f}(\,u,v,z\,) \\ \hline a\,\sqrt{\sin(\,v\,)^2+\sinh(\,u\,)^2} \\ \frac{\partial}{\partial v}\,\mathrm{f}(\,u,v,z\,) \\ \hline a\,\sqrt{\sin(\,v\,)^2+\sinh(\,u\,)^2} \\ \frac{\partial}{\partial z}\,\mathrm{f}(\,u,v,z\,) \end{array}\right]$$

5.2 Matrizen

Zur Darstellung von Matrizen wird die `array`-Umgebung verwendet. Sie ähnelt sehr stark der `tabular`-Umgebung, die für Tabellen verwendet wird. Die generelle Syntax lautet:

array-Umgebung

```
\begin{array}{ccc}       % für jede Spalte ein c (centered)
term1 & term2 & term3\\  % Terme durch & trennen, Zeilenende mit \\
term4 & term5 & term6    % in der letzten Zeile kein \\
\end{array}
```

Die `array`-Umgebung produziert ungeklammerte Matrizen. Wenn die Matrix geklammert werden soll, müssen den runden oder eckigen Klammern die Kommandos `\left` bzw. `\right` vorangestellt werden.

```
\[                      % Anfang der Formel
\left (                 % große Klammer auf
\begin{array}{cc}       % zweispaltige Matrix
x^y   & \frac{\alpha}{\beta}\\[2.5mm]
a+b+c & \frac{a+b+c}{x^2}
\end{array}             % Ende der Matrix
\right )                % große Klammer zu
\]                      % Ende der Formel
```

$$\left(\begin{array}{cc} x^y & \frac{\alpha}{\beta} \\[2.5mm] a+b+c & \frac{a+b+c}{x^2} \end{array}\right)$$

5.3 Mathematische Sonderzeichen

Die Zeichen `+ - / = ! ' | ( ) [` und `]` können ohne besondere Umstände direkt in Formeln verwendet werden. Daneben existieren unzählige weitere Sonderzeichen (Operatoren, Pfeile, mathematische Symbole), die durch LaTeX-Kommandos gebildet werden können. Die folgenden Tabellen

geben (auszugsweise) die Kommandos für die wichtigsten Sonderzeichen an. Wenn Sie diese Sonderzeichen im normalen Text (und nicht in einer Formel) verwenden möchten, müssen Sie das Kommando zwischen zwei $-Zeichen setzen!

Sonderzeichen

\infty	∞	\cdot	$\cdot$	\pm	$\pm$	\neq	$\neq$
\partial	∂	\circ	$\circ$	\times	$\times$	\sim	$\sim$
\Re	$\Re$	\bullet	$\bullet$	\div	$\div$	\simeq	$\simeq$
\Im	$\Im$	\ldots	$\ldots$	\ast	$\ast$	\approx	$\approx$
\forall	$\forall$	\vdots	$\vdots$	\\|	$\|$	\equiv	$\equiv$
\exists	$\exists$	\cdots	$\cdots$	\vee	$\vee$	\le	$\le$
		\ddots	$\ddots$	\wedge	$\wedge$	\ge	$\ge$
				\nabla	∇	\ll	$\ll$
				\oplus	$\oplus$	\gg	$\gg$
				\ominus	$\ominus$		
				\otimes	$\otimes$		

Pfeile

\leftarrow	$\leftarrow$	\Leftarrow	$\Leftarrow$	\nearrow	$\nearrow$
\rightarrow	$\rightarrow$	\Rightarrow	$\Rightarrow$	\searrow	$\searrow$
\uparrow	$\uparrow$	\Uparrow	$\Uparrow$	\swarrow	$\swarrow$
\downarrow	$\downarrow$	\Downarrow	$\Downarrow$	\nwarrow	$\nwarrow$
\leftrightarrow	$\leftrightarrow$	\Leftrightarrow	$\Leftrightarrow$		
\updownarrow	$\updownarrow$	\Updownarrow	$\Updownarrow$		
\hookleftarrow	$\hookleftarrow$				

In Formeln müssen einzelne Variablen häufig durch Vektorpfeile, Ableitungsstriche oder -punkte oder durch andere Zusatzsymbole gekennzeichnet werden. Die folgende Tabelle beschreibt die wichtigsten Markierungsbefehle. Beachten Sie bitte, dass Kommandos wie `\vec` nur für einzelne Buchstaben verwendet werden können (und nicht für Buchstabengruppen oder noch längere Ausdrücke).

Vektoren und Ableitungen

\bar{x}	$\bar{x}$	\tilde{x}	$\tilde{x}$	x'	x'	x'''	x'''
\dot{x}	$\dot{x}$	\ddot{x}	$\ddot{x}$	\vec{x}	$\vec{x}$		

5.4 Griechische und kalligrafische Buchstaben

Griechische Buchstaben werden ebenfalls durch LaTeX-Kommandos dargestellt. Der Kommandoname ergibt sich aus dem Namen des Buchstabens, also `\alpha` für ein kleines α und `\Delta` für ein großes Δ.

Griechische Buchstaben

\alpha	α	\kappa	κ	\Psi	Ψ	\Upsilon	Υ
\beta	β	\lambda	λ	\omega	ω	\varepsilon	ε
\chi	χ	\Lambda	Λ	\Omega	Ω	\varphi	φ
\delta	δ	\mu	μ	\rho	ρ	\varpi	ϖ
\Delta	Δ	\nu	ν	\sigma	σ	\varrho	ϱ
\epsilon	ϵ	\phi	ϕ	\Sigma	Σ	\varsigma	ς
\eta	η	\Phi	Φ	\tau	τ	\vartheta	ϑ
\gamma	γ	\pi	π	\theta	θ	\xi	ξ
\Gamma	Γ	\Pi	Π	\Theta	Θ	\Xi	Ξ
\iota	ι	\psi	ψ	\upsilon	υ	\zeta	ζ

Kalligrafische Buchstaben stehen nur als Großbuchstaben zur Verfügung. Zur Darstellung dieser Buchstaben müssen Sie mit `\cal` die Schriftart verändern, also beispielsweise `{\cal A} \cup {\cal B}` für $\mathcal{A} \cup \mathcal{B}$ schreiben.

6

Steuerung des Layouts

Im Großen und Ganzen führt LaTeX den Zeilen- und Seitenumbruch selbstständig durch und kommt dabei zu recht guten Ergebnissen. Es gibt aber auch Situationen, in denen LaTeX versagt. Fallweise (aber nicht immer) zeigt LaTeX Probleme beim Umbruch durch Warnungen der Form *over-/underfull hbox/vbox* an. Damit ist gemeint, dass LaTeX Text außerhalb des Seitenrands platziert hat oder dass im Text sehr große Abstände zwischen den Wörtern auftreten. Die wahrscheinlichste Fehlerursache ist ein langes Wort, in dem LaTeX keine Trennmöglichkeit erkannt hat (speziell bei Fachausdrücken).

Aber auch wenn LaTeX keine Warnungen liefert, kann es vorkommen, dass Sie mit dem Ergebnis nicht zufrieden sind – etwa weil LaTeX versucht hat, eine Seite optimal auszunutzen und dabei eine oder zwei Zeilen eines neuen Absatzes noch auf dieser Seite platziert hat, obwohl der Beginn einer neuen Seite der Übersichtlichkeit halber sinnvoller wäre. Dieser Abschnitt fasst die wichtigsten Möglichkeiten zusammen, manuell (also durch zusätzliche Kommandos) in den Umbruch einzugreifen.

6.1 Trennungen

LaTeX trennt prinzipiell automatisch. Die dabei angewandten Trennregeln sind überraschend zuverlässig. Das heißt, es kommt nur sehr selten vor, dass LaTeX wirklich falsch trennt. Sehr viel öfter tritt der Fall ein, dass LaTeX keine eindeutige Trennposition entdeckt, das Wort daher ungetrennt lässt und dann beim Zeilenumbruch in Schwierigkeiten gerät (große Löcher oder Text, der über den Seitenrand hinausragt). In diesem Fall können Sie in das betreffende Wort mit `\-` so genannte *weiche Trennungen* einfügen. LaTeX trennt das Wort dann – falls notwendig – an einer der von Ihnen angegebenen Stellen (aber dann auch nur an diesen Stellen). Am häufigsten treten Trennprobleme bei Wörtern mit deutschen Sonderzeichen auf, beispielsweise bei `an\-schlie\-ßend` oder `Schlüs\-sel\-wör\-ter`.

Manchmal tritt auch das umgekehrte Problem auf – Sie möchten vermeiden, dass LaTeX ein (oft kurzes) Wort trennt. Dazu stellen Sie einfach das gesamte Wort in ein `\mbox{}`-Kommando.

Trennung beeinflussen

`\-`	weiche Trennung
`\mbox{wort}`	Wort nicht trennen

In den LaTeX-Distributionen NTeX und teTeX sind bereits die Trenndateien für Englisch, Amerikanisch, Deutsch und einige andere Sprachen eingebaut. Die deutschen Trennungen werden durch das *german*-Package aktiviert. Wenn Ihr Text gemäß der neuen Rechtschreibung getrennt werden soll, müssen Sie statt *german* das Paket *ngerman* verwenden.

Mithilfe des Befehls `\hyphenation{Donau-dampf-schiff}` kann LaTeX mitgeteilt werden, dass das Wort (hier Donaudampfschiff) generell an den mit Minuszeichen gekennzeichneten Stellen getrennt werden darf. Es kann nicht nur ein Wort angegeben werden, sondern durch Leerzeichen getrennt eine ganze Liste von Trennvorschlägen. Leider dürfen in der Trennliste keine Wörter mit deutschen Umlauten vorkommen und die Liste darf auch nicht zu lang werden (max. 300 Wörter). Somit

ist die \hyphenation-Anweisung nur für die wichtigsten Tennungen zu verwenden. Sie sollte auch für jedes Dokument neu erstellt bzw. angepasst werden.

6.2 Wortzwischenräume und horizontale Leerräume

Um die einzelnen Zeilen eines Absatzes im Blocksatz darzustellen, fügt LaTeX zwischen allen Wörtern einen (innerhalb einer Zeile) einheitlichen Leerraum ein und nach einem Punkt (Satzende) einen etwas größeren. Wenn nach einem Punkt nur ein normaler Abstand verwendet werden soll (etwa bei Abkürzungen), muss nach dem Punkt \␣ angegeben werden: Dr.\ Huber für Dr. Huber. (␣ steht für ein Leerzeichen.) Wenn zwei Wörter nicht durch einen Zeilenumbruch getrennt werden sollen, kann ein fixes Leerzeichen mit ~ angegeben werden, beispielsweise 3~cm.

Zusätzlicher Leerraum zwischen zwei Wörtern kann durch \quad, \qquad oder \hspace{abstand} eingefügt werden. Syntaktisch überflüssige Leerzeichen vor bzw. nach diesen Kommandos sollten vermieden werden, weil sich dadurch ungewollt ein größerer Leerraum ergeben kann.

Die drei oben genannten Kommandos erzeugen Abstände einer genau vorgegebenen Breite. Ganz anders sieht die Wirkung von \hfill aus. Dieses Kommando fügt den gesamten zur Verfügung stehenden Freiraum einer Zeile an der aktuellen Position ein. eigentlich ist \hfill eine Abkürzung für \hspace{\fill}, wobei \fill als „Gummilänge“ bezeichnet wird – eben weil sie variabel ist. Wird \hfill in einer Zeile mehrfach verwendet, verkleinert sich der eingefügte Raum entsprechend. Wenn \hfill am Ende einer Zeile verwendet wird, muss die Zeile mit \hbox{} abgeschlossen werden. Damit wird ein unsichtbares LaTeX-Objekt erzeugt, das als Grenze für \hfill wirkt. Beispiel:

```
\hfill zentriert \hfill\hbox{}
```

zentriert

Zusätzliche vertikale und horizontale Abstände

\␣	Wortzwischenraum nach Interpunktionszeichen
~	fixes Leerzeichen; an dieser Stelle erfolgt kein Zeilenumbruch
\quad	zusätzlicher Leerraum der Größe 1 em (siehe unten)
\qquad	zusätzlicher Leerraum von 2 em (siehe unten)
\hspace{abstand}	Leerraum der angegebenen Größe einfügen
\hspace*{abstand}	wie oben, aber auch bei Zeilenumbruch
\hfill	so großen Abstand einfügen, dass Zeile ausgefüllt wird
\dotfill	wie oben, aber statt Leerraum punktierte Linie
\hrulefill	wie oben, aber durchgezogene Linie
\hbox{}	unsichtbares LaTeX-Objekt (als Abgrenzung für \hfill)

6.3 Zeilenumbruch und vertikale Leerräume

Innerhalb eines Absatzes beginnt LaTeX generell nur dann eine neue Zeile, wenn in der aktuellen Zeile kein Platz mehr ist. Einen vorzeitigen Zeilenwechsel können Sie mit \\ erreichen. Wenn Sie dahinter in eckigen Klammern einen Abstand angeben, fügt LaTeX außerdem einen entsprechenden vertikalen Leerraum ein: \\[1cm]. Wenn zwischen \\ und der Maßangabe ein * steht, wird der Leerraum auch dann eingefügt, wenn ein Seitenwechsel durchgeführt wird. (Das ist nur in den seltensten Fällen sinnvoll!) Zwischen zwei Absätzen kann mit \vspace{abstand} ein zusätzlicher Abstand eingefügt werden. Für alle Maße sind auch negative Zahlen erlaubt. Auch bei \vspace kann man die Gummilänge \fill verwenden. \vspace{\fill} oder kurz \vfill füllt die Seite bis zum nächsten \newpage-Befehl. Man kann so beispielsweise Seiten erzeugen, die nur oben und unten etwas Text enthalten und dazwischen leer sind.

Manueller Zeilenumbruch

`\\`	Zeilenwechsel ohne Randausgleich (kein Blocksatz in dieser Zeile)
`\\[abstand]`	Zeilenwechsel mit erhöhtem Abstand zur nächsten Zeile
`\\*[abstand]`	wie oben, aber Abstand auch bei Seitenwechsel
	Zeilenwechsel mit Randausgleich (Blocksatz)
`\vspace{abstand}`	zusätzlicher Abstand zwischen zwei Absätzen
`\vspace*{abstand}`	wie oben, aber Abstand auch bei Seitenwechsel

6.4 Fester Seitenumbruch

Ein fester Seitenumbruch kann mit den drei Kommandos `\newpage`, `\pagebreak` und `\clearpage` erreicht werden. Die Unterschiede gehen aus der folgenden Tabelle hervor.

Manueller Seitenumbruch

`\newpage`	beginnt eine neue Seite/Spalte, der Rest der Seite bleibt leer
`\pagebreak`	wie oben mit vertikalem Randausgleich (größerer Absatzabstand)
`\clearpage`	neue Seite, auch bei zweispaltigem Text

6.5 Eigene Kopfzeilen

Normalerweise kümmert sich LaTeX selbstständig um Kopfzeilen und gestaltet diese je nach Texttyp. Am aufwändigsten geht LaTeX dabei beim Texttyp *book* vor. Es unterscheidet zwischen geraden und ungeraden Seiten und nimmt die Kapitel- und Abschnittsüberschrift mit in die Kopfzeilen auf. Man nennt dies „lebende Kolumnentitel“. Wenn Sie mit den Kopf- und Fußzeilen nicht zufrieden sind, können Sie über `\pagestyle` die automatischen Kopfzeilen deaktivieren und via `\markright` oder `\markboth` eigene Kopfzeilen definieren. LaTeX fügt den so definierten Kopfzeilen automatisch die laufende Seitenzahl hinzu.

Kopfzeilen

`\pagestyle{headings}`	automatische Gestaltung der Kopfzeile (Default-Einstellung)
`\pagestyle{empty}`	keine Kopfzeile, keine Seitenzahl
`\pagestyle{plain}`	keine Kopfzeile, Seitenzahl zentriert in der Fußzeile
`\pagestyle{myheadings}`	eigene Kopfzeile, siehe `\markright` und `\markboth`
`\thispagestyle`	wie `\pagestyle`, aber nur für eine Seite
`\markright{Kopfzeile}`	Kopfzeile für einseitige Texte
`\markboth{links}{rechts}`	Kopfzeile für zweiseitige Texte

Mit den hier beschriebenen Kommandos ist es nicht möglich, die optische Gestaltung der automatischen LaTeX-Kopfzeilen zu verändern. LaTeX stellt die Überschriften in den Kopfzeilen standardmäßig in Großbuchstaben und ohne Unterstreichung dar, also ganz anders als in diesem Buch. Wenn ein anderes Layout gewünscht wird, muss die Einstellung durch eine eigene Style-Datei verändert werden. Noch eleganter geht es, wenn die Style-Datei *`fancyhdr.sty`* zur Verfügung steht (`{\usepackage{fancyhdr}`). Ein Beispiel, wie man die LaTeX-typischen Kopfzeilen los wird:

```
...
\pagestyle{fancy}
\renewcommand{\chaptermark}[1]{\markboth{\thechapter\ #1}{}}
\renewcommand{\sectionmark}[1]{\markright{\thesection\ #1}}
\fancyhf{}
\fancyhead[LE,RO]{\thepage}
\fancyhead[LO]{\nouppercase \rightmark}
\fancyhead[RE]{\nouppercase \leftmark}
...
```

Damit sind die Kopfzeilen klein und die Seitennummer steht immer ganz aussen. Das ist schlicht und einfach, sieht aber gut aus.

6.6 Globale Layouteinstellung

Das Layout eines LaTeX-Textes kann nicht nur durch diverse, zumeist nur auf kleine Textausschnitte angewandte Kommandos beeinflusst werden, sondern auch durch globale Einstellungen. Diese Einstellungen werden normalerweise vor dem eigentlichen Beginn des Textes (also vor `\begin{document}`) vorgenommen und gelten dann für den gesamten Text. Viele der im Folgenden aufgelisteten Kommandos können aber auch irgendwo im Text verwendet werden und entfalten ihre Wirkung dann erst ab dieser Stelle.

Zur Veränderung der meisten Maßangaben existieren eigene Kommandos. Dabei gilt folgende Syntax: `\setlength\kommando{`*maß*`}`. Einige Einstellungen müssen durch eine direkte Veränderung von LaTeX-Variablen durchgeführt werden. In diesem Fall lautet die Syntax: `\setcounter{name}{wert}`.

Die folgende Tabelle fasst die Maße zur Einstellung der bedruckten Bereiche einer Seite zusammen. Die Kommandos sind dabei logisch von links nach rechts bzw. von oben nach unten geordnet. Im Bild 6.1 wird das noch deutlicher.

Seitenlayout	
`\setlength{\oddsidemargin}{` *maß* `}`	Abstand linker Papierrand zum Text (ungerade Seiten)
`\setlength{\evensidemargin}{` *maß* `}`	Abstand linker Papierrand zum Text (gerade Seiten)
`\setlength{\textwidth}{` *maß* `}`	Textbreite (bedruckter Bereich)
`\setlength{\topmargin}{` *maß* `}`	Abstand oberer Papierrand zur Kopfzeile
`\setlength{\headheight}{` *maß* `}`	Höhe der Kopfzeile
`\setlength{\headsep}{` *maß* `}`	Abstand Kopfzeile – Text
`\setlength{\textheight}{` *maß* `}`	Texthöhe (für den eigentlichen Text ohne Kopf- und Fußzeilen)
`\setlength{\columnsep}{` *maß* `}`	Spaltenabstand (nur bei zweispaltigem Text)
`\setlength{\columnseprule}{` *maß* `}`	Stärke der Linie zwischen den Spalten (default 0)
`\setlength{\footskip}{` *maß* `}`	Abstand zwischen Text und dem unteren (!) Ende der Fußzeile
`\setlength{\footheight}{` *maß* `}`	Höhe der Fußzeile

Standardmäßig fügt LaTeX zwischen zwei Absätzen keinen Zwischenraum ein. Dafür wird ab dem zweiten Absatz eines Abschnitts die jeweils erste Zeile des Absatzes ein wenig eingerückt. Wenn Sie eine Absatzgestaltung wie in diesem Buch erreichen möchten (kein Einzug, dafür spürbarer Absatzabstand), müssen Sie `\parindent` auf 0 setzen (Maßeinheit nicht vergessen – trotz der 0).

Normalerweise macht LaTeX keinen zusätzlichen vertikalen Abstand zwischen den einzelnen Absätzen. Man kann aber durch Aufruf der Befehle `\smallskip`, `\medskip` oder `\bigskip` dafür sorgen, einen kleinen vertikalen Abstand (viertel, halbe oder ganze Zeile) zwischen allen Absätzen einzufügen. Insofern gehören die Befehle in die Style-Datei oder in den Dokumentenvorspann. Ein Abstand beliebiger Höhe kann durch das Kommando `\renewcommand{\parskip{höhe}}` festgelegt werden.

Erklärungsbedürftig ist auch das Kommando `\flushbuttom`: Es bewirkt, dass LaTeX einen vertikalen Randausgleich durchführt. Dabei wird zwischen Absätzen und Überschriften so viel Leerraum eingefügt, dass alle Seiten exakt gleich lang sind. Standardmäßig ist diese Formatierung nur beim Texttyp *book* aktiviert und kann dort mit `\raggedbottom` abgeschaltet werden.

Absatzlayout

`\parindent`*maß*	Einzug der ersten Zeile des Absatzes
`\bigskip`	einzeiliger Abstand zwischen Absätzen
`\medskip`	halbzeiliger Abstand zwischen Absätzen
`\smallskip`	viertelzeiliger Abstand zwischen Absätzen
`\parskip`*maß*	Abstand zwischen zwei Absätzen
`\raggedright`	kein Blocksatz, sondern Flattersatz
`\flushbottom`	vertikaler Randausgleich (gleich bleibende Seitenhöhe)
`\raggedbottom`	kein vertikaler Randausgleich

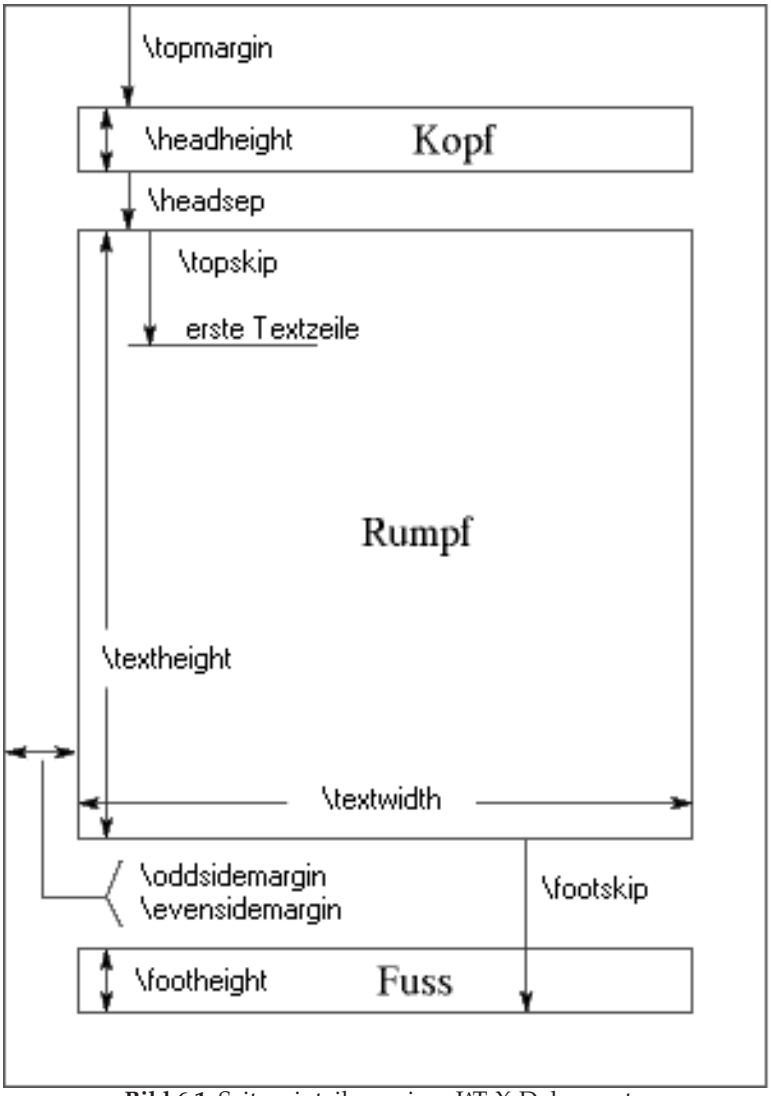

Bild 6.1: Seiteneinteilung eines LaTeX-Dokuments

Die Syntax zur Veränderung des Zeilenabstands weicht von der der obigen Kommandos ab: Mit `\renewcommand{\baselinestretch`*faktor*`}` wird der normale Zeilenabstand um den angegebenen Faktor vergrößert oder verkleinert. Diese etwas ungewöhnliche Definition ist notwendig, weil der Zeilenabstand von der aktuellen Schriftgröße abhängig ist und daher nicht auf einen starren Wert gesetzt werden sollte.

Üblicherweise nummeriert LaTeX alle Überschriften bis einschließlich `\subsubsection` automatisch durch, was zu Abschnittsnummern wie 3.5.2.1 führt. In diesem Buch wurde die Nummerierung durch eine Veränderung der Variablen *secnumdepth* auf zwei Ebenen (Kapitel und Abschnitte) reduziert. Analog kann die Anzahl der Ebenen für das Inhaltsverzeichnis über die Variable *tocdepth* gesteuert werden (in diesem Buch drei Ebenen).

Nummerierung von Überschriften

`\setcounter{secnumdepth}{n}`	*n*+1 Gliederungsebenen nummerieren
`\setcounter{tocdepth}{n}`	*n*+1 Gliederungsebenen ins Inhaltsverzeichnis
`\setcounter{page}{n}`	verändert die laufende Seitenzahl
`\setcounter{chapter}{n}`	verändert die laufende Kapitelzahl
`\setcounter{section}{n}`	verändert die laufende Abschnittszahl

6.7 Farben

In LaTeX können Sie Farben nach dem RGB (rot grün blau)-Farbmodell (additive Farbmischung, z. B. beim Monitor), CMY(K) (cyan magenta yellow (black))-Farbmodell (subtraktive Farbmischung, z. B. beim Drucker) oder HSB (hue saturation brightness) sowie in Graustufen definieren. Es wird das Paket *color* benötigt.

LaTeX-Farbmodelle

Name	*Basisfarben*	*Wertebereich*
rgb	red, green, blue	0.0 . . . 1.0
cmy	cyan, magenta, yellow	0.0 . . . 1.0
cmyk	cyan, magenta, yellow, black	0.0 . . . 1.0
hsb	hue, saturation, brightness	0.0 . . . 1.0
gray	gray	0.0 . . . 1.0
RGB	Red, Green, Blue	0 . . . 255
HSB	Hue, Saturation, Brightness	0 . . . 255
Gray	gray	0 . . . 15

Bei *RGB, HSB* und *Gray* sind nur ganze Zahlen erlaubt, bei allen anderen Modellen die üblichen Zahlen mit Kommastellen. Außerdem gibt es noch das Modell *named* mit vordefinierten Farben (s. u.), die aber nicht jeder DVI-Treiber akzeptiert.

TeX-Farben sind in der Regel nach dem CMYK-Modell definiert, weil der Drucker das typische Ausgabegerät ist. LaTeX greift auf die Datei `color.pro` zu, in der diese Definitionen stehen, außerdem wird `color.pro` ggf. von DVIPS-Treibern gelesen (Wobei die Linux-Version von `xdvi` keine Farben darstellt, Sie müssen die DVI-Datei erst in eine PS-Datei umwandeln, um die Farben zu sehen.) Diese Farben treffen – bis auf wenige Ausnahmen – annähernd Farbtöne aus der PANTONE-Palette. PANTONE ist der Standard in der Druckindustrie. Die Farbauszeichnung nach dem PMS (Pantone Matching System) sollte daher auf verschiedenen Druckern ein in etwa gleiches Druckergebnis gewährleisten.

Bei Folien (siehe nächsten Abschnitt) liegt der Fall etwas anders, hier kann man sich auf das RGB-Modell stützen, bei dem die Farben aus den Anteilen Rot, Grün und Blau gemischt werden – wie beim Fernsehen oder auch bei den Farbangaben von HTML-Seiten. Die Definition der Farben erfolgt mit dem Befehl `\definecolor{Farbname}{Modell}{Werte}`, die Umschaltung auf die Farbe dann mit `\color{Farbname}`. Das Verhalten ist hier wie beim Fettdruck oder der Schriftgröße – entweder man schaltet die Farbe dauerhaft jeweils um oder man schließt den einzufärbenden Bereich in geschweifte Klammern ein. Beispiele für Farbdefinitionen:

```
% Modell RGB, Werte für Rot-, Grün- und Blau-Anteil, durch Komman getrennt
\definecolor{white}{rgb}{1,1,1}
\definecolor{black}{rgb}{0,0,0}
\definecolor{gold}{rgb}{1.0,0.84,0}
\definecolor{pastellgruen}{rgb}{0.855,1.0,0.90}
\definecolor{pastellblau}{rgb}{0.85,0.95,1.0}
\definecolor{pastellpink}{rgb}{0.965,0.785,0.89}
\definecolor{pastellgelb}{rgb}{0.875,1.0,0.75}

Hier ist ein Wort in {\color{pastellpink} pastellpink} eingefärbt.
```

Immer verfügbar sind die vordefinierten Farben mit den Namen *black, white, red, green, blue, cyan, magenta* und *yellow*.

Bei der Anwendung von `dvips` werden Sie feststellen, dass nicht alle Farben gleich gut wirken. Manche Pastelltöne kommen zu schwach oder auch zu kräftig. Auch fällt manchem die Namenswahl schwer. Daher sind in der Datei *`dvipsnam.def`* etliche Farbnamen vordefiniert. Mit folgendem LaTeX-Dokument können Sie sich eine Farbtabelle erzeugen (das neu definierte Kommando `SC` zeigt die jeweilige Farbe als rechteckiges Feld und ihren Namen).

```
\documentclass[12pt,a4paper]{article}
\usepackage[usenames,dvipsnames]{color}
\usepackage{multicol}
\pagestyle{empty}
\setlength{\parindent}{0pt}

\newcommand{\SC}[1]{%
\textcolor[named]{#1}{\rule{10mm}{10mm}}\quad
\texttt{#1}\strut\\}

\begin{document}
\noindent
\begin{multicols}{3}
\SC{GreenYellow}   \SC{Yellow}       \SC{Goldenrod}
\SC{Dandelion}     \SC{Apricot}      \SC{Peach}
\SC{Melon}         \SC{YellowOrange} \SC{Orange}
\SC{BurntOrange}   \SC{Bittersweet}  \SC{RedOrange}
\SC{Mahogany}      \SC{Maroon}       \SC{BrickRed}
\SC{Red}           \SC{OrangeRed}    \SC{RubineRed}
\SC{WildStrawberry}\SC{Salmon}       \SC{CarnationPink}
\SC{Magenta}       \SC{VioletRed}    \SC{Rhodamine}
\SC{Mulberry}      \SC{RedViolet}    \SC{Fuchsia}
\SC{Lavender}      \SC{Thistle}      \SC{Orchid}
\SC{DarkOrchid}    \SC{Purple}       \SC{Plum}
\SC{Violet}        \SC{RoyalPurple}  \SC{BlueViolet}
\SC{Periwinkle}    \SC{CadetBlue}    \SC{CornflowerBlue}
\SC{MidnightBlue}  \SC{NavyBlue}     \SC{RoyalBlue}
\SC{Blue}          \SC{Cerulean}     \SC{Cyan}
\SC{ProcessBlue}   \SC{SkyBlue}      \SC{Turquoise}
\SC{TealBlue}      \SC{Aquamarine}   \SC{BlueGreen}
\SC{Emerald}       \SC{JungleGreen}  \SC{SeaGreen}
\SC{Green}         \SC{ForestGreen}  \SC{PineGreen}
\SC{LimeGreen}     \SC{YellowGreen}  \SC{SpringGreen}
\SC{OliveGreen}    \SC{RawSienna}    \SC{Sepia}
\SC{Brown}         \SC{Tan}          \SC{Gray}
\SC{Black}         \SC{White}
\end{multicols}
\end{document}
```

Um die Farben zu nutzen, gibt es mehrere Möglichkeiten. Neben dem Befehl `\color` gibt es für kurze Textstücke `\textcolor{Farbe}{Text}`, der einzelne Wörter oder Passagen einschließt.

Auf die gleiche Weise lässt sich auch die Hintergrundfarbe des Textes modifizieren. `\pagecolor{Name}` bzw. `\pagecolor [modell] {spezifikation}` setzt die Hintergrundfarbe der Seite. Dieser Befehl wirkt immer auf die gesamte Seite und bis zum Ende des Dokuments. Um eine einzelne Seite farbig zu hinterlegen, eignet sich die Befehlsfolge

```
\pagecolor{yellow}%
\afterpage{\pagecolor{white}}
```

Der Befehl `\normalcolor` dient zum Zurücksetzen der Farbeinstellungen auf die Anfangswerte.

Neben der Einfärbung von Schrift und Seitenhintergrund gibt es die Möglichkeit, den Hintergrund einer (Text)Box farbig zu gestalten. Der Befehl `\colorbox{Name}{Text}` hinterlegt den Text mit einer Farbbox ohne Rahmen. Statt des Farbnamens kann auch die Kombination `[Modell]{Spezifikation}` angegeben werden. Die Box hat einen Rand der Breite `\fboxsep`. Wenn Sie einen zusätzlichen Rahmen wollen, verwenden Sie den Befehl `\fcolorbox{R}{H}{Text}`, der wie `\fbox` funktioniert. *R* spezifiziert die Rahmenfarbe (Name) und *H* die Hintergrundfarbe. `\fboxrule` steuert die Rahmenbreite. Dazu ein Beispiel:

```
\colorbox{gray}{Ein grau unterlegter Text}

\setlength{\fboxsep}{5mm}
\setlength{\fboxrule}{3mm}
\fcolorbox{red}{blue}{Ein nichtssagender Text}
```

Das ergibt (hier sind aus drucktechnischen Gründen auch Blau und Rot als Graustufen dargestellt worden):

Ein grau unterlegter Text

Color-Befehle

Befehl	Bedeutung
`\color{Farbe}`	setzt die Textfarbe auf die angegebene Farbe
`\textcolor{Farbe}{Text}`	desgleichen für kurze Textstücke
`\normalcolor`	detzt die Farbe auf den Defaultwert
`\pagecolor{Name}`	einfärben des Seitenhintergrunds
`\colorbox{Name} {Text}`	farbiges Hinterlegen des Textes (So wie bei diesem Kasten)
`\fcolorbox{R}{H}{Text}`	text farbig hinterlegen mit Farbe *H*; zusätzlich ein Rahmen der Farbe *R*

Bei den meisten Befehlen ist statt eines Farbnamens auch die Angabe von Modell und Farbspezifikation (`\befehl[modell]{spezifikation}` möglich.

6.8 Texte rotieren

Manchmal möchte man Texte nicht in der normalen Richtung anordnen, sondern senkrecht oder in irgendeinem Winkel gegenüber der Waagrechten. Da hilft das Paket `rotating`. Es stellt drei neue Umgebungen zur Verfügung, `rotate`, `turn` und `sideways`. Alle bewirken die Rotation des eingefassten Textes um den als Argument angegebenen Winkel. Der Winkel wird als Zahl ohne Einheit angegeben, LaTeX fasst dies als Angabe in (Alt-)Grad auf, gemessen gegen den Uhrzeigersinn von der Horizontalen aus.

- Alles innerhalb der `sideways`-Umgebung wird um 90 Grad entgegen dem Uhrzeigersinn gedreht.

  ```
  \begin{sideways}
    ...
  \end{sideways}
  ```

  Für eingebundene Bilder gibt es `sidewaysfigure`, das anstelle von `figure` verwendet wird. Dazu passend dreht `rotcaption` die Bildunterschrift.

- Alles innerhalb der `rotate`-Umgebung wird um einen bestimmten Winkel gedreht. Dabei wird dem gedrehten Text kein Platz eingeräumt. Man muss daher mit geeigneten Mitteln für genügend Raum sorgen (z. B. mittels Boxen). Es lassen sich damit sehr gut senkrechte Beschriftungen bei Objekten variabler Größe anbringen, wie beispielsweise in den Informationskästen hier im Buch.

  ```
  \begin{rotate}{Winkel}
    ...
  \end{rotate}
  ```

- Bei `turn` wird ausgehend von der linken unteren Ecke im Uhrzeigersinn um `Winkel` Grad gedreht. Anders als bei `rotate` wird dem gedrehten Text genügend Platz gelassen.

  ```
  \begin{turn}{Winkel}
    ...
  \end{turn}
  ```

Das folgende Beispiel dient als erste Spielerei:

```
\begin{document}
\begin{turn}{45}\fbox{Um 45° nach oben geneigter Text.}\end{turn}
\begin{turn}{315}\fbox{Um 45° nach unten geneigter Text.}\end{turn}
\begin{turn}{405}\fbox{Um 405° (=360+45°) geneigert Text.}\end{turn}
\begin{turn}{180}\fbox{Extra für die Australier.}\end{turn}
\end{document}
```

Gut geeignet ist die Rotation, wenn man schmale Tabellen setzen will, die aber relativ lange Spaltenlegenden haben. Dazu ein Beispiel:

```
\begin{tabular}{lccc}
\rule{0mm}{20mm} % Platz für Legende frei halten
\begin{rotate}{60}Betriebssystem\end{rotate}\hfill &
\begin{rotate}{60}MS-DOS\end{rotate} &
\begin{rotate}{60}Windows\end{rotate} &
\begin{rotate}{60}Linux\end{rotate} \\
\hline
Multiuser & - & - & x \\
Multitasking & - & x & x \\
\hline
\end{tabular}
```

Das Beispiel liefert als Ergebnis:

Betriebssystem	MS-DOS	Windows	Linux
Multiuser	-	-	x
Multitasking	-	x	x

7

Gestaltung wissenschaftlicher Texte

Prinzipiell können Sie jeden beliebigen Text mit LaTeX setzen. Seine Vorteile gegenüber anderen Satz- und Textverarbeitungsprogrammen spielt LaTeX aber erst bei der Gestaltung wissenschaftlicher Texte oder bei Büchern aus, bei denen es darum geht, mit geringem Aufwand Inhalts-, Abbildungs-, Literatur- und Stichwortverzeichnisse zu erstellen, Querverweise zu verwenden, Fußnoten einzufügen etc.

7.1 Die Titelseite

Die Makros der Titelseite hängen sehr von dem gewählten Style ab und sind eigentlich nur bei „book“ oder „article“ interessant. Für die Gestaltung der Titelseite bieten sich folgende Makros an:

`\title`: Titel des Dokumentes
`\author`: Autor des Dokumentes
`\date`: Datum der Veröffentlichung

Alle diese Befehle sind optional und müssen vor dem eigentlichen Dokument (`\begin{document}`) angegeben werden. Um die Titelseite auszugeben, wird dann der Befehl `\maketitle` verwendet. Beim Befehl `\date` kann entweder `\today` für das aktuelle Datum, eine beliebige Datumsangabe oder gar nichts angegeben werden.

Die Zusammenfassung (Abstract) wird in der Umgebung `abstract` angegeben und steht dann unter dem Titel. Diese Umgebung steht nicht in der Dokumentenklasse „book“ zur Verfügung. Hier ein Beispiel für den Vorspann zu einem Artikel:

```
\documentclass{article}
\usepackage[latin1]{inputenc}

\author{X. B. Liebig und Jürgen Plate}
\title{Linux für Studenten}
\date{23.03.2006}

\begin{document}
\maketitle

\begin{abstract}
Diese Abhandlung beschäftigt sich mit der Aufzucht und
Pflege kleiner, elektronischer Pinguine. Als Umgebung
haben sich {\bf Linux} und \LaTeX\ bewährt. In Umgebungen
mit zu vielen Windows neigen die kleinen TUXe zu allergischen
Reaktionen.
\end{abstract}

...

\end{document}
```

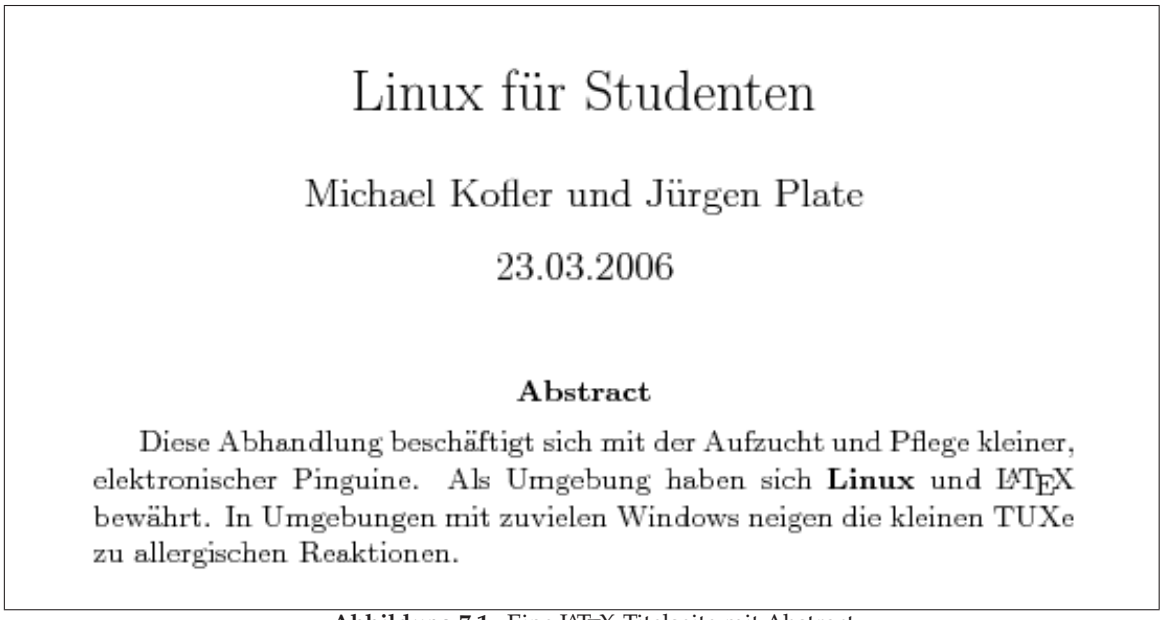

Abbildung 7.1: Eine LaTeX-Titelseite mit Abstract

Das Ergebnis ist in Abbildung 7.1 zu bewundern:

Titelseite	
`\author`	definiert den Autor
`\title`	definiert den Titel
`\date`	definiert das Veröffentlichungsdatum
`\maketitle`	erzeugt die Titelseite im Dokument
`abstract`	Umgebung für die Zusammenfassung

7.2 Bearbeitung umfangreicher Texte

Wenn Sie längere Texte mit LaTeX schreiben möchten, ist es sinnvoll, den Text in mehrere Dateien zu zerlegen. Bewährt hat sich dabei die Unterteilung in eine zentrale Steuerungsdatei (z. B. *buch.tex*), eine Datei mit globalen Definitionen und Einstellungen (Stylefile, z. B. *buch.sty*) sowie je eine Datei für jedes Kapitel. (LaTeX hat übrigens auch bei solcherart zusammengesetzten Dokumenten keinerlei Probleme mit dem Inhalts- und Stichwortverzeichnis. Querverweise können ohne weiteres von einem Kapitel in ein anderes verweisen.)

Das wichtigste Kommando zum Zusammensetzen von Texten aus mehreren Dateien lautet `\input{datei}`. Es liest an der Stelle seines Auftretens die angegebene Datei und verarbeitet sie, als stünde der darin enthaltene Text an der Stelle des `\input`-Kommandos. `\input` darf auch verschachtelt auftreten. Wenn in `\input` keine Dateikennung angegeben wird, hängt LaTeX selbstständig *.tex* an den Dateinamen an.

externe Datei einlesen	
`\input{datei}`	Fügt die angegebene Datei als Teil des Quelltextes an der Stelle des `input`-Befehls ein. Im Gegensatz zum `include`-Befehl erfolgt kein Seitenvorschub.
`\include{datei}`	Erzeugt ein `\newpage` und fügt danach die angegebene Datei an der Stelle des `include`-Befehls ein

Die Zerlegung langer Texte in mehrere Dateien hat nicht nur den Vorteil einer besseren Übersichtlichkeit, sie beschleunigt auch das Arbeitstempo. Während der Arbeit an einem Kapitel können alle anderen `\include`-Zeilen in *buch.tex* durch ein vorangestelltes %-Zeichen auskommentiert werden.

Die Bearbeitung des Textes durch LaTeX wird dadurch erheblich beschleunigt. Die Dateien *buch.tex* und *buch.sty* sehen in der Regel etwa so aus:

```
% buch.tex: zentrale Steuerungsdatei
\documentclass{book}
\usepackage[fleqn]{amsmath}
\usepackage{amssymb}
\usepackage{alltt}
\usepackage{latexsym}
\usepackage{makeidx}
\usepackage{textcomp}

\usepackage{buch}          % eigenes Zeug

\makeindex
\begin{document}

\include{Text/preface}
\tableofcontents
\newpage
\include{Text/kap1}
\include{Text/kap2}
...
\include{Text/kap19}
\printindex
\end{document}
```

Das Stylefile (Endung `.sty`) enthält ganz normale LaTeX-Anweisungen, aber auch Definitionen von Makros und Umgebungen – eben alles, was man so braucht. Es beginnt mit der Zeile `\ProvidesPackage{Name des Pakets}[Erstellungsdatum]`. Die `RequirePackage`-Zeilen legen fest, welche Pakete ggf. benötigt werden.

```
\ProvidesPackage{buch}[2005/08/31]

\RequirePackage{german}
\RequirePackage[latin1]{inputenc}
\RequirePackage{verbatim}
\RequirePackage{eurosym}       % Euro-Symbol

% ------------------------------------ Allgemeine Formatanweisungen
\setlength{\parskip}{4pt}% Einzug am Begin des Absatzes
\parindent 0pt         % verzichte auf Einrücken der ersten Zeile

\widowpenalty=10000   %Schusterjungen und Hurenkinder möglichst
\clubpenalty=10000    %ausschließen
\sloppy               %Anzahl der Warnungen reduzieren (es bleiben genug!)

...
```

7.3 Inhaltsverzeichnis

Das Inhaltsverzeichnis kann mit dem Kommando `\tableofcontents` an jeder beliebigen Stelle in den Text eingefügt werden. Üblicherweise wird das Inhaltsverzeichnis am Anfang des Texts oder nach dem Vorwort platziert.

Zur Erstellung des Inhaltsverzeichnisses verarbeitet LaTeX die Datei *name.toc*. In diese Datei trägt LaTeX bei jedem Durchlauf mit dem Kommando `\tableofcontents` alle Informationen für das Inhaltsverzeichnis ein (Kapitel-, Abschnitts- und Teilabschnittsnamen zusammen mit ihren Seitennummern). Diese Vorgehensweise hat zur Folge, dass LaTeX im ungünstigsten Fall *dreimal* ausgeführt werden muss, bis das Inhaltsverzeichnis korrekt ist.

Beim ersten Mal existiert *name.toc* noch nicht, d. h. es kann kein Inhaltsverzeichnis erzeugt werden. Beim zweiten Mal kann zwar ein Inhaltsverzeichnis angelegt werden, da dieses aber zumeist ebenfalls einige Seiten beansprucht, verschieben sich alle Seitennummern! Erst beim dritten Durchlauf stimmen die Seitennummern im Inhaltsverzeichnis mit den tatsächlichen Seitennummern überein.

In das Inhaltsverzeichnis werden normalerweise die Texte aller vorhandenen `\part`-, `\chapter`-, `\section`-, `\section`- und `\subsection`-Kommandos aufgenommen. Die Anzahl der Gliederungsebenen kann mit `\setcounter{tocdepth}{n}` vermindert werden (siehe Seite 55).

Die Formatierung des Inhaltsverzeichnisses erfolgt automatisch. LaTeX wählt dabei passende Schriftgrößen, rückt untergeordnete Einträge ein und füllt den Zeilenfreiraum zwischen dem Eintrag und der Seitennummer mit Punkten. Das Inhaltsverzeichnis dieses Buchs ist kein typisches Beispiel, weil hier das Standardlayout von LaTeX aus ästhetischen Gründen verändert wurde. Eine bessere Vorstellung, wie ein Inhaltsverzeichnis von LaTeX normalerweise aussieht, bietet die Abbildung 1.2 auf Seite 16.

Inhaltsverzeichnis

`\tableofcontents`	fügt an dieser Stelle das Inhaltsverzeichnis ein

7.4 Querverweise

Querverweise im Buch werden mit den drei Kommandos `\label`, `\ref` und `\pageref` erstellt. Welche Nummer `\ref` liefert, hängt von dem Ort ab, an dem `\label` ausgeführt wurde: In der Regel handelt es sich um eine Abschnittsnummer, die den aktuellen Abschnitt bezeichnet. `\label` kann aber auch in Umgebungen wie `equation`, `figure` oder `table` verwendet werden. `\ref` liefert in diesem Fall die Nummer der Formel, Abbildung, Tabelle etc.

Mit Querverweisen verhält es sich ähnlich wie mit den Seitenzahlen des Inhaltsverzeichnisses: Sie werden der Datei *name.aux* entnommen, die beim letzten LaTeX-Durchlauf erstellt wurde. Nach Änderungen in einem LaTeX-Text sind daher wie oben mindestens zwei Durchläufe notwendig, bis alle Seitennummern korrekt sind.

Abschließend ein Beispiel: Wenn an dieser Stelle im Text ein Markierungspunkt mit `\label{verweis-bsp}` erstellt wird, dann liefert `\ref{verweis-bsp}` das Ergebnis 7.4 und `\pageref{verweis-bsp}` die Seitennummer 64.

Querverweise

`\label{marke}`	definiert einen Markierungspunkt
`\pageref{marke}`	liefert die Seitennummer der Markierung
`\ref{marke}`	liefert Abschnitts-, Abbildungs- oder Tabellennummer

7.5 Fußnoten

Fußnoten werden mit `\footnote{text}` erstellt. LaTeX fügt daraufhin an dieser Stelle eine hochgestellte Nummer als Fußnote ein und platziert den Fußnotentext an das Ende der laufenden Seite.

Fußnoten werden automatisch durchnummeriert, wobei die Nummerierung bei den Texttypen *book* und *report* in jedem Kapitel neu beginnt. Das Aussehen von Fußnoten ist in Abbildung 1.2 auf Seite 16 dokumentiert.

Fußnoten

`\footnote{text}`	fügt eine neue Fußnote in den Text ein

7.6 Der Anhang

Der Anhang wird mit `\begin{appendix}` eingeleitet und mit `\end{appendix}` beendet. Sie haben hier die gleichen Möglichkeiten der Gestaltung wie im Dokumententext, im Anhang werden die Kapitel jedoch alphabetisch nummeriert.

Anhang

```
\begin{appendix}

 ...        % alle Kapitel des Anhangs

 \end{appendix}
```

7.7 Literaturverzeichnis

Die Verwaltung des Literaturverzeichnisses erfolgt in zwei Schritten: Zuerst muss am Ende des Buchs (dort, wo das Literaturverzeichnis erscheinen soll) eine Liste mit allen Einträgen erstellt werden. Anschließend kann im gesamten Text auf die Einträge dieses Verzeichnisses verwiesen werden.

Literaturverzeichnis

```
% im laufenden Text
\cite{marke1}          % Verweis auf den Eintrag 'marke1'

% am Ende des Artikels/Buchs
\begin{thebibliography}{n}
\bibitem{marke1} Text1  % Autor, Titel etc.
\bibitem{marke2} Text2
\end{thebibliography}
```

LaTeX erzeugt an der Stelle der `thebibliography`-Umgebung eine Liste, in der alle Einträge in eckigen Klammern durchnummeriert werden ([1], [2] etc.). Der Parameter *n* gilt als Maß für die Einrückung der Einträge. Für bis zu neun Einträge kann für *n* die Zahl 9 eingesetzt werden, für bis zu 99 Einträge die Zahl 99 etc. Die `\bibitem`-Einträge werden nicht automatisch sortiert – Sie müssen sich also selbst um eine geeignete Ordnung kümmern. Die Einträge werden auch nicht automatisch formatiert. Sie können aber alle Kommandos zur Einstellung der Schriftart verwenden und so den Namen des Autors fett, den Titel kursiv etc. formatieren.

Im laufenden Text können Sie nun mit `\cite{marke}` auf einen Eintrag im Literaturverzeichnis verweisen. LaTeX setzt an dieser Stelle im Text eine eckige Klammer mit der entsprechenden Nummer ein.

Während das `\bibitem`-Kommando für gelegentliche Publikationen vollkommen ausreichend ist, bietet LaTeX eine noch viel leistungsstärkere Alternative durch das Zusatzprogramm *bibtex*. Die vollständige Liste aller jemals (etwa in einer ganzen Abteilung) verwendeten Referenzen wird in einer eigenen Datei gespeichert. In der eigentlichen Publikation kann darauf durch Kürzel verwiesen werden. *bibtex* wertet diese Informationen aus und erstellt automatisch ein Quellenverzeichnis, das nur die tatsächlich genutzten Referenzen enthält. Dieses Zusatzprogramm ist im LaTeX-Begleiter von Michel Goosens et al. ausführlich beschrieben.

Das Quellenverzeichnis dieses Buchs wurde übrigens nicht mit den gerade beschriebenen Methoden erstellt. Die wenigen Quellen erfordern keine Nummerierung und die meisten Titel beziehen sich auf weiterführende oder vertiefende Literatur.

7.8 Stichwortverzeichnis

Wenn Sie Ihren Text mit einem Stichwortverzeichnis ausstatten möchten, dann sind dazu mehrere Arbeitsschritte erforderlich: Sie müssen mit `\usepackage` das Package *makeidx* laden und vor `\begin{document}` das Kommando `\makeindex` ausführen. Die Indexeinträge müssen im Text mit dem Kommando `\index` markiert werden. Schließlich muss an der Stelle im Text, an der das Stichwortverzeichnis erscheinen soll, das Kommando `\printindex` angegeben werden.

Doch damit nicht genug: Nachdem die so präparierte Textdatei zum ersten Mal mit LaTeX bearbeitet wurde, muss das Linux-Kommando `makeindex name.idx` ausgeführt werden. Dabei wird die von LaTeX erstellte Datei *`name.idx`* mit allen Indexeinträgen ausgewertet und sortiert (mit der Option `-g` sogar nach der deutschen Sortierordnung). Das Ergebnis wird in der Datei *`name.ind`* gespeichert. Beim nächsten LaTeX-Durchlauf wird diese Datei bei der Ausführung von `\printindex` eingelesen.

Die wichtigsten Syntaxvarianten von `\index{eintrag}` gehen aus der folgenden Tabelle hervor. Mit den Zeichenkombinationen `|(` und `|)` lassen sich Seitenbereiche angeben. Der resultierende Indexeintrag sieht dann beispielsweise so aus: Eintrag 34-38. Ein nach `|` und *ohne* vorangestelltes \-Zeichen angegebenes Kommando kann zur Formatierung der Seitenzahl eingesetzt werden. Ein Muster für die Definition eines geeigneten Kommandos wurde mit `\ii` angegeben. `@` kann dazu eingesetzt werden, auch Formeln oder speziell sortierte Einträge richtig im Indexverzeichnis einzuordnen. Alle genannten Formatierungsmethoden können auch kombiniert werden, beispielsweise um einen Subeintrag besonders hervorzuheben.

Einträge in das Stichwortverzeichnis

```
\usepackage{makeidx}                % nach \documentclass
...
\makeindex                          % vor \begin{document}
\newcommand{\ii}[1]{{\it #1}}       % kursive Seitenzahlen
...
\index{Eintrag}                     % normaler Indexeintrag
\index{Haupteintrag!Subeintrag}     % Subeintrag
\index{Haupt!Sub!Subsub}            % Subsubeintrag
\index{Eintrag|(}                   % Eintrag von Seite
\index{Eintrag|)}                   % Eintrag bis Seite
\index{Eintrag|ii}                  % Seitennummer kursiv
\index{Pi@$\pi$}                    % Formel wie 'Pi' sortieren
\index{Eintrag@{\bf Eintrag}}       % fett, richtig  sortieren
...
\printindex                         % vor \end{document}
```

Abschließend folgen noch einige Beispiele, die die Wirkungsweise des `\index`-Kommandos demonstrieren. Die Ergebnisse können Sie sich im Stichwortverzeichnis am Ende des Buchs ansehen.

```
\index{Indexbeispiel}
\index{Indexbeispiel!Subeintrag}
\index{Indexbeispiel!Subeintrag@{\tt Subeintrag Courier}}
\index{Indexbeispiel!Sonderz@{\verb?Sonderzeichen % \?}}
\index{Indexbeispiel!kursiver Eintrag@{\it kursiver Eintrag}}
\index{Indexbeispiel!kursive Seitenziffer|ii}
\index{index@{\verb?\index?}}
\index{printindex@{\verb?\printindex?}}
\index{makeindex@{\verb?makeindex?}}
\index{usepackage@\verb?\usepackage?!makeidx@{\tt makeidx}}
\index{Stichwortverzeichnis!LaTeX}
\index{Index}
```

Zum Programm `makeindex` existiert eine detaillierte Beschreibung als `man`-Seite. Der Text hat allerdings den Nachteil, dass er nicht LaTeX-spezifisch ist. (`makeindex` kann auch zur Bearbeitung von Indexdateien anderer Programme eingesetzt werden.) Eine typische Steuerdatei für das Programm sieht folgendermaßen aus:

```
preamble
 "\\begin{theindex}\n
  \\thispagestyle{empty}\n"
postamble
 "\\end{theindex}\n"
delim_0 "\\hspace{1mm} \\dotfill "
delim_1 ", "
delim_2 ", "
delim_r "-"
delim_t
item_0 "\n \\item "
item_1 "\n \\subitem "
item_2 "\n \\subsubitem "
item_01 "\n \\subitem "
item_x1 "\n \\subitem "
item_12 "\n \\subsubitem "
item_x2 "\n \\subsubitem "
indent_space
indent_length 0
group_skip "\n\n \\indexspace\n"
encap_prefix "\\"
encap_infix "{"
encap_suffix "}"
headings_flag 1
symhead_positive "Symbole"
symhead_negative "symbole"
numhead_positive "Ziffern"
numhead_negative "ziffern"
heading_prefix "\n \\vspace{2pt} {\\Large "
heading_suffix "} \\nopagebreak \\vspace{1pt} \n"
```

Wenn die Steuerdatei *`myind.sty`* heißt und die LaTeX-Datei *`linuxbuch.tex`*, wird der Index mit `makeindex -s myind.sty linuxbuch` generiert.

8

Briefe schreiben

Da haben Sie sich bei diversen Projekten mit LaTeX angefreundet und nun wollen Sie einen Brief schreiben – und Sie überlegen, wie lange das wohl dauert, bis Sie das mit allen Feldern (Anschrift, Absender, Anlagen, Faltmarken usw.) hingefrickelt haben. Vielleicht hat Sie ein Versuch mit der Umgebung `letter` auch nicht so ganz befriedigt. Für LaTeX existiert glücklicherweise die `dinbrief`-Umgebung, die recht brauchbar ist. Sie macht z. B. standardkonforme Briefe mit Markierungen zum Falten am linken Rand und der Empfängeradresse genau an der Stelle für Fensterumschläge usw.

Ein Dokument kann mehrere Briefe enthalten, die jeweils innerhalb einer `dinbrief`-Umgebung angegeben werden. Eine entscheidende Bedeutung beim Schreiben von Briefen kommt dem `\opening`-Kommando zu: nur dieser Befehl setzt den Briefkopf, die Absenderangaben und die Empfängeradresse. Darauf folgt der eigentliche Brieftext. Abschließend steht der `\closing`-Befehl, der mit zusätzlichen optionalen Argumenten eine Unterschrift als Text oder Grafik einbindet.

Mithilfe der Dokumentenklasse *dinbrief* oder der entsprechenden KOMA-Script-Variante kann man auf einfache Art und Weise einen DIN-gerechten Brief verfassen. Wer zu faul zum Lesen der Dokumentation ist, findet hier eine passende Vorlage. Die einzelnen Elemente erklären sich nahezu von selbst:

```
\documentclass[norm]{dinbrief}
\usepackage{german}
\usepackage[latin1]{inputenc}

\begin{document}
\pagestyle{empty}          % keine Seitennummern sonst {plain}

%
\Absender{\textbf{Hans Mustermann\\
          Musterstr. 7\\[2ex]
          12345 Musterhausen}}
%
% Adresse etwas über die Linie heben
\backaddress {\raisebox{0.6mm}{Hans Mustermann
               $\bullet$ Musterstr. 7
               $\bullet$ 12345 Musterhausen}}
%
\signature{Hans Mustermann}
%
\Datum{\today}
\place{Musterhausen}
%
%--- Empfänger
\begin{letter}{Name\\
               Straße \par
               PLZ Ort}
%
%--- Betreff
\subject {\textbf{Betreff}}
%
\nowinwowrules % keine Linien um die Adresse
```

```
%--- Anrede
\opening{Sehr geehrte Damen und Herren,}

%--- Brieftext


Hier der Brieftext\\[3cm]

blabla bla blabla bla bla\\
blabla bla blabla bla bla\\[3cm]


%
%--- Briefende
\closing{Mit freundlichen Grüßen}
%
%--- Anlagen
% --> ggf. löschen/auskommentieren
\encl{Diverse Anlagen}
%
\end{letter}
\end{document}
```

Die Befehlsliste für den Brief ist nicht sehr umfangreich und schnell zu lernen. Wenn man sich dann noch ein Shellskript schreibt, das den Lieblingseditor mit einer Kopie der Briefvorlage aufruft und auch gleich die Übersetzung und den Druck erledigt, geht so ein Brief auch mit LaTeX ganz fix.

dinbrief-Befehle

Befehl	Bedeutung
`\address{}`	Name und Adresse des Absenders
`\stdaddress{}`	Absenderadresse nach DIN 5008
`\signature{}`	Unterschrift des Absenders
`\backaddress{}`	Absenderadresse im Brieffenster
`\place{}`	Ortsangabe im Brief (Dingenskirchen, den ...)
`\date{}`	Briefdatum
`\yourmail{}`	Ihre Zeichen, Ihre Nachricht vom
`\sign{}`	Unsere Zeichen (, unsere Nachricht vom)
`\phone{}`	Rufnummer
`\writer{}`	Sachbearbeiter
`\subject{}`	1. Betreff
`\concern{}`	2. Betreff
`\opening{}`	Anrede
`\closing{}`	Grußformel
`\encl{}`	Anlagen
`\ps{}`	Postscriptum
`\cc{}`	Verteiler
`\bottomtext{}`	Konto, Handelsregister usw.
`\postremark{}`	Postvermerk
`\handling{}`	Behandlungsvermerk
`\centeraddress`	Absenderadresse zentrieren
`\normaladdress`	Absenderadresse linksbündig
`\nowindowrules`	keine Linien um die Empfängeradresse
`\windowrules`	Linien um die Empfängeradresse
`\nowindowtics`	keine Faltmarken
`\windowtics`	Faltmarken

Markus Kohm, der federführende Autor des KOMA-Script-Pakets, will mit seinem Paket Bedürfnisse einer europäischen Typografie bedienen. Eigens für die Erstellung von Briefen hat Kohm eine Klasse *scrlttr2* geschaffen. Das Paket ersetzt die Vorgängerversion *scrlettr*. Für eine eingehendere Beschäftigung mit *scrlttr2* verweisen wir auf die deutschsprachige Dokumentation (texmf/doc/latex/koma-script/scrguide.pdf).

Es lassen sich auch sehr leicht Serienbriefe schreiben. Sie benötigen dazu nur ein kleines Makro wie das folgende (wie man Makros selbst schreibt, erfahren Sie im folgenden Abschnitt):

```
\newcommand{\Brief}[1]{
  \begin{letter}{#1}
  \input{Brieftext}
  \end{letter}}
```

Mit dem Befehl `\input{Brieftext}` wird der Text des Serienbriefs aus der Datei *Brieftext* geladen, die natürlich wieder alle LaTeX-Elemente von *dinbrief* enthalten kann. Das Makro packen Sie mit in das Dokument:

```
\documentclass[norm]{dinbrief}
\usepackage{german}
\usepackage[latin1]{inputenc}

\newcommand{\Brief}[1]{
  \begin{letter}{#1}
  \input{Brieftext}
  \end{letter}}

\begin{document}
% Vorspann wie oben
  ...
% nur statt \begin{letter} kommen nun die Makroaufrufe
\Brief{Thusnelda Bims\\
       Lange Gasse 123\\
       98765 Sonstwo}
\Brief{Waldemar von Adlig\\
       Königstr. 12\\
       45678 Burghausen}
  ...
%
\end{letter}
\end{document}
```

Auch die Makro-Aufrufe mit den Adressen können Sie in eine zweite Datei auslagern, die dann per `\input`-Befehl gelesen wird. Eine solche Daten kann auch leicht per Programm oder Datenbankabfrage generiert werden.

Auch lässt sich das neu definierte Makro noch ausblasen, um beispielsweise unterschiedliche Anreden (Herr, Frau, …) zu generieren. Außerdem gibt es fertige LaTeX-Pakete für Serienbriefe, unter anderem „mailing“, „textmerg“, „finder“ und „formletter“.

9

Folien und Präsentationen erstellen

Für Präsentationen wird heuzutage in der Regel Powerpoint verwendet und man kann schon froh sein, wenn der Vortragende dabei gewisse Grundregeln einhält, etwa des Verwenden großer Schrift und das Darbieten von nicht zuviel Informationen. Steht dem Vortragenden kein Beamer zur Verfügung, werden die PP-Seiten einfach auf Folien gedruckt. Dabei gilt oft: Form dominiert über Funktion – d. h. Logos, Rahmen, Farbverläufe usw. lassen die Informationen beinahe nicht mehr erkennen. Dafür dauert es Stunden, bis die Folie fertig ist. Sie können aber auch LaTeX verwenden, um Folien zu erstellen, und dabei unter etlichen, teilweise recht mächtigen Paketen wählen. Wir wollen hier zwei Pakete vorstellen: `seminar`, ein einfaches Paket zum Erstellen von Vortragsfolien, das man schon als „Großvater aller Präsentationen“ bezeichen könnte, und einen Abkömmling, `beamer`, der als PDF-Präsentation viele interaktive Möglichkeiten bietet.

9.1 Folien erstellen mit Seminar

Das Paket ist in der Regel schon in der LaTeX-Installation enthalten. Falls nicht, erhalten Sie es unter *http://www.tug.org/applications/Seminar/*. Das Paket wurde von Timothy Van Zandt entwickelt und zeichnet sich wie gesagt durch ein einfaches Layout aus. Es bietet nur wenige Features, z. B. fehlt das schrittweise Einblenden von Informationen.

Das Seminar-Paket unterscheidet zwei Formate von Folien, das Querformat (landscape) und das Hochformat (portrait). Als Grundstil dient die Dokumentenklasse *seminar*. Einzelne Folien werden im normalen Text mit `\begin{slide}` und `\end{slide}` umschlossen. Es werden automatisch größere Schriften etc. verwendet. Der Quellcode für eine super-einfache Landscape-Folie lautet:

```
\documentclass[11pt]{seminar}
% Fontgroessen-Voreinstellung 11pt; möglich sind 11pt oder 12pt
\usepackage{german}
\usepackage[latin1]{inputenc}
\begin{document}
\begin{slide}
Eine Folie im Landscape-Format.
\end{slide}
\end{document}
```

Die Fontgrößen-Angabe entspricht dem, was man bei `article` oder `book` verwenden würde, die real verwendeten Fonts sind jedoch größer). Beim Betrachten mit GhostScript kommt keine Freude auf, es scheint so, als ginge irgendetwas schief – das Papierformat ist im Portrait-Modus, die Folie im Landscape-Modus. Beim Drucken im Landscape-Modus kommen die Folien aber richtig aus dem Gerät. Dem Manko können Sie mit einem kleinen Makro (siehe folgenden Abschnitt) abhelfen. Die Header müssen dann natürlich mitrotiert werden:

```
% folgender Befehl richtet sich nicht an LaTeX,
% sondern wird durchgereicht an dvips
% Seitenorientierung richtig einstellen
\renewcommand{\printlandscape}{\special{landscape}}
\rotateheaderstrue
```

Innerhalb einer `slide`-Umgebung kann mit `\newslide` eine neue Folie erzwungen werden. Normalerweise wird nach `\end{slide}` automatisch eine neue Folie begonnen. Mit dem Befehl `\extraslideheight{len}` kann der Seitenwechsel gesteuert werden. Wenn Sie beispielsweise `\extraslideheight{100cm}` verwenden, müssen Sie den Seitenumbruch mit `\newslide` erzwingen. `\extraslideheight{0pt}` überlässt LaTeX den Umbruch ohne Kompromisse. Die Voreinstellung ist `\extraslideheight{10pt}`.

9.1.1 Orientierung

Verwendet man `slide*` anstelle von `slide`, wird die Orientierung gewechselt (portait statt landscape). Man kann auch durch die Option `portrait` bei `\documentclass` gleich das Hochformat als Standard einstellen (auch dann muss `slide*` für Folien im Hochformat verwendet werden). Durch die Befehle `landscapeonly` und `portraitonly` kann man die Ausgabe so steuern, dass jeweils nur Folien der entsprechenden Orientierung ausgegeben werden.

Das Seminar-Paket bietet den Befehl `\slidesmag{n}` an, um den Folieninhalt zu vergrößern oder verkleinern. Die Größe wird als 1.2^n berechnet. n muss dabei eine ganze Zahl zwischen -5 und 9 sein.

Das Format der Folie kann mit den drei folgenden Befehlen verändert werden:

- `\slidewidth{breite}` gibt die Breite der Folien an.
- `\slideheight{höhe}` gibt die Höhe der Folien an.
- Die Parameter `Breite` und `Höhe` bei `\begin{slide}[Breite,Höhe]` erlauben bei jeder Folie eine Änderung der Größe.
- `\centerslidestrue` (Voreinstellung) zentriert die Folie auf dem Papier, mit `\centerslidesfalse` wird nicht zentriert.
- `\raggedslides[len]` erlaubt es, den Randausgleich am rechten Rand zu steuern. Normalerweise erfolgt Flattersatz (Voreinstellung). Der Wert *len* steuert den Ausgleich, mit `\raggedslides[opt]` bekommen Sie Randausgleich.

Natürlich lassen sich wie bei anderen Dokumentenstilen die Ränder mittels `\slideleftmargin`, `\sliderightmargin`, `\slidetopmargin`, `\slidebottommargin`, `\paperwidth` und `\paperheight` individuell einstellen (siehe Dokumentation zum Paket).

9.1.2 Schriftarten

Die Standardschrift mit Serifen eignet sich wunderbar für Bücher und Artikel, aber weniger für Folien, wo eine serifenlose Schrift besser und klarer zur Geltung kommt. Durch die Option `semhelv` beim `\documentstyle` kann auf die serifenlose Helvetica-Schrift umgeschaltet werden (ggf. ist noch ein `\usepackage{semhelv}` im Vorspann nötig). Die Schriften im Mathematik-Modus bleiben, wie sie waren. Alternativ kann auch mit der Option `semlcmss` die serifenlose Schrift aus den Computer Modern Fonts eingebunden werden.

9.1.3 Ränder und Rahmen

Der Befehl `\slideframe[options]{style}` ermöglicht es, die Ränder der Folien zu gestalten. Mögliche Stile sind `none` oder `plain`. Die `pagestyles` `none`, `plain`, `headings` oder `myheadings` verhalten sich genauso wie bei `article`. Mit den beiden Zeilen

```
\pagestyle{headings}
\markright{Folien mit \textbf{seminar.cls}}
```

erhalten Sie beispielsweise eine Überschrift („Folien ...“) mit Seitennummer auf jeder Folie. Sollen Kopf- und Fußzeilen individuell gestaltet werden, hilft ein eigener Seitenstil, z. B.:

```
\newpagestyle{meiner}%
   {Folien mit \textbf{seminar.cls} \hfill \rightmark \hfill \thepage}%
   {X. B. Liebig, Jürgen Plate \hfill  \today}%
\pagestyle{meiner}
```

Hier erscheinen die Überschrift links oben, die Seitennummer rechts oben, die Autoren links unten und das Datum rechts unten. Sie könnten natürlich auch noch ein kleines Logo einbinden oder das Ganze noch aufpeppen. mit `\renewpagestyle` kann der Seitenstil auch jederzeit umdefiniert werden.

Weitere Stile können beispielsweise mit `\documentstyle [fancybox]{seminar}` eingebunden werden, darunter *shadow* (Folienumrandung mit Schatten), *double* (doppelte Folienumrandung), *oval* (ovale Folienumrandung) oder *Oval* (dickere ovale Folienumrandung). Es muss dann aber das Paket *fancybox* eingebunden werden. Zum Beispiel:

```
\documentstyle[12pt,fancybox,semhelv]{seminar}
\usepackage{german}
\usepackage[latin1]{inputenc}
\usepackage{fancybox}
\renewcommand{\printlandscape}{\special{landscape}}
\slideframewidth 0.5cm
\slideframe{oval}
\begin{document}
\begin{slide}
Folie mit ovalem Rand.
\end{slide}
\end{document}
```

`\slideframewidth` legt die Dicke der Rahmenlinie fest (Voreinstellung 4pt) und `\slideframesep` den Abstand zwischen Rahmen und Inhalt (Voreinstellung 0.4in).

Übrigens können Sie natürlich auch die zugehörigen Boxen `\shadowbox{Text}`, `\ovalbox{Text}`, `\Ovalbox{Text}` und `\doublebox{Text}` zur Gestaltung Ihrer Folien verwenden.

9.1.4 Farben

Folien nur in Schwarz und Weiss sind sicher etwas langweilig. Farbe bringt Aufmerksamkeit und kann zur Gliederung des Dargebotenen beitragen. Neben den schon auf Seite 56 behandelten allgemeinen Möglichkeiten zur Einfärbung des Textes werden bei der Seminar-Dokumentenklasse mit `\usepackage{semcolor}` einfache Voraussetzungen für Farben geschaffen. Sie können auch hier beliebige eigene Farben definieren, wobei Ihnen alle Farbmodelle zur Verfügung stehen. Wir bleiben mal bei nur einem Modell, das den meisten am geläufigsten sein dürfte, dem RGB-Modell. Zum Definieren von Farben dient hier der Befehl `\newrgbcolor{Name}{Rotanteil Grünanteil Blauanteil}`, z. B.:

```
\newrgbcolor{gray}{0.9 0.9 0.9}
\newrgbcolor{lred}{1 0.5 0.5}
\newrgbcolor{orange}{1.0 0.5 0.0}
\newrgbcolor{blue}{0.0 0.0 1.0}
```

Damit kann man dann loslegen; die Farbnamen definieren die Textfarbe, mit geschweiften Klammern wird der Gültigkeitsbereich festgelegt. Mittels `\colorbox{Farbe}{Inhalt}` können Sie auch farbig hinterlegte Texte erzeugen:

```
{\gray Dies ist} {\lred eine Folie}
{\orange mit bunten} {\blue Farben}.
...
\colorbox{gray}{Ein grau unterlegter Text}
```

9.2 Präsentationen erstellen mit Beamer

Auch dies Paket ist in der Regel schon in der LaTeX-Installation enthalten und bietet alles, was das Herz begehrt (soweit es um das Erstellen von Präsentationen geht). Die Folien sind per default auf die Größe 128 mm x 96 mm, also ein 4:3-Format festgelegt. Das Postkartenformat wird durch den PDF-Betrachter auf Bildschirmgröße skaliert. Es kann bei der Schriftgröße zwischen `11pt` und `12pt` gewählt werden. Navigationselemente werden automatisch erzeugt.

Einige vom Paket benötigten Pakete, darunter *color*, *xcolor* und *hyperref* werden automatisch geladen.

Das Vorgehen beim Produzieren der fertigen Dateien ist etwas anders als bisher (Sie können das übrigens beim Seminar-Paket auch so machen):

- Erstellen der Folien wie unten beschrieben → `folien.tex`
- PDF-Präsentation erstellen: `pdflatex folien.tex` → `folien.pdf`
- Falls Bilder eingebunden werden, müssen diese nicht im PS- oder EPS-Format vorliegen, sondern als JPG, PDF, PNG oder TIFF.
- Einige Pakete erfordern die Angaben von `pdflatex` als Option, beispielsweise `includepackage[pdftex]{color}` oder `includepackage[pdftex]{hyperref}`.

Anstelle des Kommandos `latex folien.tex` erspart man sich hier den Umweg über eine dvi- und eine PostScript-Datei und generiert mittels `pdflatex folien.tex` direkt die PDF-Datei für die Präsentation. Mit dem Acrobat Reader kann man dann die Präsentation im Vollbild-Modus laufen lassen und mit den Cursortasten zwischen den Folien wechseln. Das Programm `pdflatex` ist Bestandteil fast aller LaTeX-Distributionen.

Auch das Beamer-Dokument beginnt mit einigen Standardbefehlen, unter anderem mit der Auswahl der Dokumentenklasse und des Themas bzw. Stils der Präsentation. Beamer bietet zahlreiche Stile, die alle nach Städten benannt sind. Voreingestellt ist `default`. Die Stile heißen:

Beamer-Farbschemata

Antibes	Bergen	Berkeley	Berlin
Boadilla	Copenhagen	Darmstadt	Dresden
Frankfurt	Goettingen	Hannover	Ilmenau
JuanLesPins	Luebeck	Madrid	Malmoe
Marburg	Montpellier	PaloAlto	Pittsburgh
Rochester	Singapore	Szeged	Warsaw

Bei der Dokumentenklasse kann als optionaler Parameter die Grundfarbe der Präsentation angegeben werden. Wählt man hier eine bestimmte Farbe, z. B. `[red]`, wird die Präsentation in diversen Rottönen gestaltet, bei `[blue]` in Blautönen usw.

Das Grundgerüst eines Beamer-Vortrags stellt sich somit folgendermaßen dar:

```
\documentclass{beamer}
\usepackage{german}
\usepackage[latin1]{inputenc}

\usetheme{Malmoe}   % oder irgend ein anderes

\begin{document}
\title{\LaTeX\ Beamer Class}
\author{Jürgen Plate}
\date{\today}
\maketitle % oder auch \frame{\titlepage}

% Hier ggf. \frame{\tableofcontents}

\begin{frame}
  \frametitle{Einleitung}
  \framesubtitle{Wie Rotkäppchen in den Wald kam}
    ...
\end{frame}

\end{document}
```

9.2.1 Abschnitte

Das Beamer-Interface ist genauso einfach wie das Seminar-Paket. Jede Folie wird in eine `\frame`-Umgebung eingebettet. Außerdem kann auf die `\section`- und `\section`-Struktur zurückgegriffen werden (wie bei der Dokumentenklasse `article`). Die Struktur wird in den Kopf der Folien übernommen. Mit dem Befehl `\frame{\tableofcontents}` wird eine Übersichtsseite erzeugt, deren Elemente auch der Navigation dienen. Für mehrteilige Präsentationen, z. B. mehrtägige Schulungen, ist auch eine übergeordnete Gliederung mit dem Befehl `\part` möglich.

Für kurze Frames kann anstelle der `\frame`-Umgebung auch ein Befehl `\frame{...}` verwendet werden. Der Befehl `\verb` ist in Frames nur dann erlaubt, wenn beim `\begin{frame}`-Befehl

der optionale Parameter `[containsverbatim]` hinzugefügt wird. Weitere wichtige Optionen sind `[plain]`, welche die Navigationselemente, Kopf- und Fußzeilen unterdrückt. Das wird manchmal benötigt, um seitenfüllend ein Bild zu präsentieren. Als Bezugspunkt für die Navigation kann `[label=name]` eingefügt werden, was eine Wiederholung dieser Folie mit `\againframe` möglich macht.

9.2.2 Blocks und Kästen

Eine weitere Strukturierungsmöglichkeit der Folie bieten abgesetzte Blöcke, die dann je nach gewähltem Layout gestaltet werden (mit/ohne Rahmen, farbig hinterlegt usw.):

```
\begin{frame}
  \frametitle{Titel}
  \begin{block}{Titel des Blocks}
    Hier steht dann der tolle Text zum ersten Block...
  \end{block}
  \begin{block}{Titel des 2. Blocks}
    Und hier der nicht minder wichtige Text zum 2. Kasten...
  \end{block}
\end{frame}
```

Dabei ist der erste Parameter der *block*-Umgebung die Überschrift des Kastens, die farbig hervorgehoben wird. Auf die gleiche Weise funktioniert der `\alertblock`, bei dem die Überschrift in der vordefinierten Farbe für Alerts (siehe später) hervorgehoben wird.

Ähnliche Umgebungen gibt es für Lehrsätze und alles was dazugehört: `definition`, `example`, `theorem`, `corollary`, `fact`, `proof` und `lemma`.

Ein Frame kann in mehrere Spalten aufgeteilt werden. Der optionale Parameter in eckigen Klammern gibt an, wie die beiden Spalten zueinander vertikal ausgerichtet werden: **b**: nach der untersten Zeile, **c**: zentriert und **t**: nach der obersten Zeile. Man spart sich so bei Gegenüberstellungen das Gefrickel mit `tabbing` oder `tabular` bzw. `minipage`. Beispiel:

```
\begin{columns}[t]
 \begin{column}{2cm}
 Erste Zeile \\
 Zweite Zeile
 \end{column}
 \begin{column}{2cm}
 Nur eine Zeile
 \end{column}
\end{columns}
```

9.2.3 Overlays

Der einfachste Weg, um Overlays zu erzeugen, ist der `\pause`-Befehl. Sie schreiben einfach den `\pause`-Befehl an die Stelle, bis zu der die Folie am Anfang sichtbar sein soll und dann vor jeden weiteren Schritt. Ein Beispiel:

```
\begin{frame}
  \frametitle{Einstieg}
  \pause
  \begin{itemize}
  \item Warum?
      ...
  \pause
  \item Darum!
      ...
  \pause
  \item Zum Schluss:
      ...
   \end{itemize}
\end{frame}
```

Die einzelnen Punkte erscheinen jeweils nach und nach, der Rest der Folie bleibt, wie er ist. Aber das ist nur die einfachste Möglichkeit. Beim Beamer-Paket lassen sich die tollsten Sachen machen. Dazu gibt es spezielle Overlay-Spezifikationen und etliche neue Befehle. Wer jetzt gerade erst mit LaTeX warm geworden ist, wird überrascht sein, denn die Overlay-Spezifikationen werden weder in geschweifte noch in eckige Klammern eingeschlossen, sondern in Kleiner- und Größerzeichen. Also

aufpassen! Innerhalb der Klammerung wird festgelegt, auf welchen Folien das jeweilige Element der Folie (genauer: welche LaTeX-Gruppe) erscheinen soll.

Innerhalb einzelner Folien kann man mit den Befehlen `\only` und `\visible` in Kombination mit einer numerischen Angabe eine Animationsreihenfolge erzeugen. Diese Reihenfolge wird durch mehrere Ausgabeseiten pro Folie realisiert, sodass die Präsentationen in jedem PDF-Viewer komfortabel abgespielt werden können. Dazu ein Beispiel:

```
\begin{frame}
  \visible<1>Erste Folie!
  \visible<2->Ab der zweiten Folie.
  \only<3>Nur auf der dritten Folie.
\end{frame}
```

Der Unterschied zwischen `\only` und `\visible` besteht darin, dass `\only` den Text tatsächlich nur auf der Seite erzeugt, für die er vorgesehen ist. `\visible` erzeugt dagegen den Text immer und „druckt" ihn auf den nichtspezifizierten Seiten mit der Hintergrundfarbe. Daher nehmen `\only`-Elemente auf den übrigen Seiten keinen Platz ein, während `\visible`-Elemente immer Platz beanspruchen (und damit die übrigen Elemente verschieben). In der Regel sorgt letzteres für einen ruhigeren Folienaufbau, da sich der Rest der Darstellung nicht verschiebt.

Overlay-Spezifikationen

$< n >$	Anzeige nur auf Folie n
$< n- >$	Anzeige ab Folie n
$< -n >$	Anzeige nur bis Folie n
$< n-m >$	Anzeige nur auf Folien n bis m
$< i, j-k, n, m-p >$	Beliebige Kombination der obigen Optionen, durch Kommata getrennt

Eine Angabe wie $< 4- >$ bewirkt also, dass das entsprechende Element auf der vierten und allen folgenden Seiten der jeweiligen Folie sichtbar ist. $< 3-5, 8 >$ bedeutet, dass das Element nur auf den Seiten 3, 4, 5 und 8 zu sehen ist. Bei etlichen Konstruktionen von LaTeX ist der `\visible`-Effekt schon eingebaut, z. B. bei Aufzählungen. Um nacheinander Punkte erscheinen zu lassen, geben Sie bei jedem Punkt an, auf welchen Folien er zu sehen ist:

```
\begin{itemize}
  \item<1-> Erster Punkt, auf jeder Folie.
  \item<2-> Ab der zweiten Folie.
  \item<3-> Ab der dritten Folie.
  \item<4-> Und ab der vierten Folie.
\end{itemize}
```

Die `\item`-Elemente unterstützen also numerische Angaben ohne zusätzliches Kommando. Wenn Sie im Beispiel oben nach dem ersten Punkt einen weiteren einschieben, müssen Sie lästigerweise alle folgenden Optionen ändern. Bei einfachen Aufzählungen wie im Beispiel, bei denen die einzelnen Punkte in der Folienreihenfolge aufgedeckt werden, gibt es dafür Abhilfe. Sie setzen einfach hinter jeden `\item`-Befehl die Spezifikation $< +- >$. Das Pluszeichen steht für den internen Item-Zähler. Es geht aber noch einfacher, indem man die Spezifikation für die gesamte Aufzählung festlegt:

```
\begin{itemize}[<+->]
  \item Erster Punkt, auf jeder Folie.
  \item Ab der zweiten Folie.
  ...
\end{itemize}
```

Der Befehl `\alert` erlaubt die Hervorhebung einzelner Folienelemente (normalerweise durch rote Textfarbe). Indem man diesen Befehl mit einer numerischen Angabe kombiniert, lassen sich Elemente an bestimmten Stellen der Präsentation hervorheben:

```
\begin{itemize}
  \item<1-> \alert<1> {Erster Punkt, auf jeder Folie.}
  \item<2-> \alert<2> {Ab der zweiten Folie.}
  \item<3-> \alert<3> {Ab der dritten Folie.}
  ...
\end{itemize}
```

Mithilfe des `\action`-Befehls lassen sich mehrere Kommandos und Seitenangaben verbinden. Auf diese Weise lassen sich z. B. Elemente alternieren und zugleich hervorheben:

```
\begin{itemize}
  \item<+-| alert@+> Erster Punkt, auf jeder Folie.
  \item<+-| alert@+> Ab der zweiten Folie.
  \item<+-| alert@+> Ab der dritten Folie.
  ...
\end{itemize}
```

Beachten Sie das Leerzeichen hinter dem senkrechten Strich. Bei `alert@` können natürlich auch wieder alle möglichen Overlay-Spezifikationen hinter dem Klammeraffen stehen. Anstelle von *alert* sind auch noch folgende Aktionen möglich:

uncover: Anzeige des Items oder Blocks (Standardaktion)
only: Anzeige nur auf den spezifizierten Folien
visible: Text wird nur bei den spezifizierten Folien gezeigt.
invisible: Text wird nicht bei den spezifizierten Folien gezeigt.

Die Overlay-Spezifikationen können nicht nur bei Aufzählungen stehen, Sie können beispielsweise die folgenden Dinge realisieren:

```
\begin{itemize}
  % Fettdruck oder andere Hervorhebungen
  \textbf<2-4>{Dieser Text erscheint auf den Folien zwei,
               drei und vier fettgedruckt.}
  % Farbe
  \color<3->{green}{Dieser Text grün ab der dritten Folie.}
  % oder einfach nur eine LaTeX-Gruppe
  \begin{block}{Dieser Text ist ab Folie zwei zu sehen}<2->\end{block}

  ...
\end{itemize}
```

Beachten Sie aber, dass die Aufzählungsbefehle (`itemize` bzw. `enumerate`) beim Beamer-Paket nicht mehr identisch mit den Originalbefehlen sind (Bild 9.1).

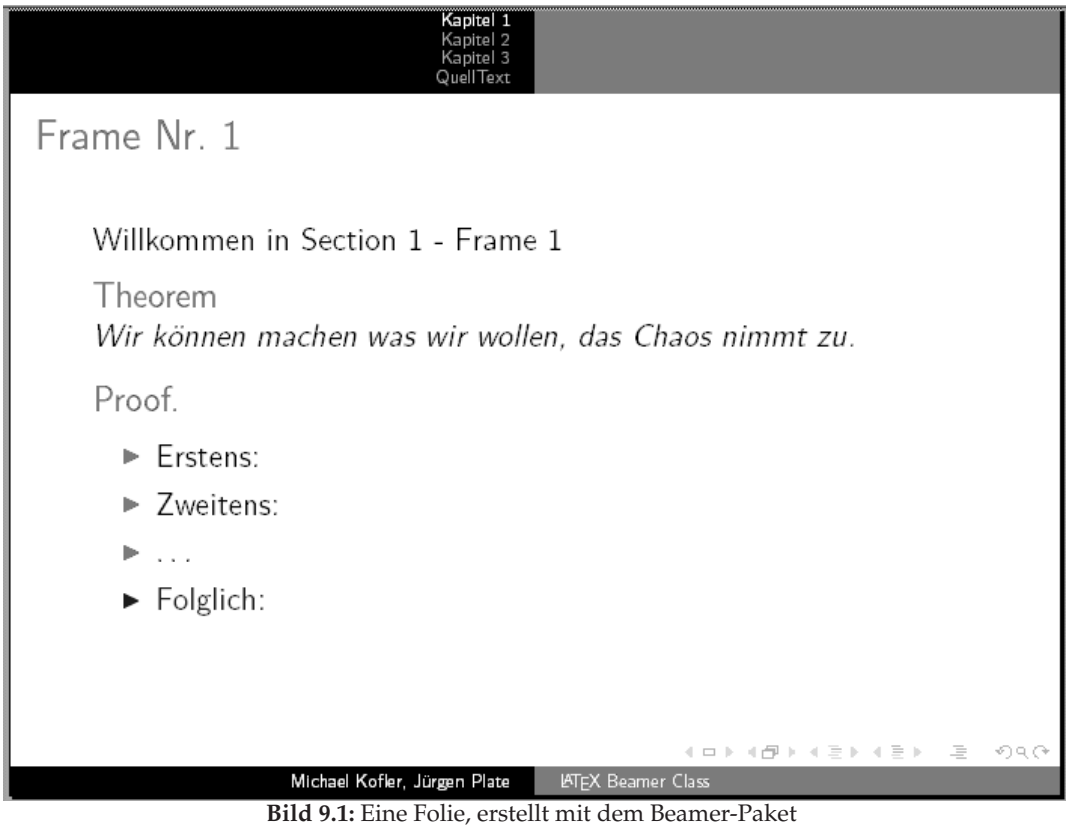

Bild 9.1: Eine Folie, erstellt mit dem Beamer-Paket

9.2.4 Hyperlinks

Im Beamer-Paket besteht – analog zu HTML – die Möglichkeit, explizite Links zu generieren. Hierzu werden Stellen im Dokument, auf die Sie referenzieren möchte, mit einer „Verankerung“ versehen. Dies geschieht folgendermaßen:

```
\hypertarget{Zielname}{beschreibender Text}
```

Auf die so erzeugten Verankerungen kann man an jeder Stelle des Dokumentes mittels

```
\hyperlink{Zielname}{beschreibender Text}
```

verweisen. Vollautomatisch werden von `pdflatex` intern Hypertargets für Einträge im Inhaltsverzeichnis gesetzt. Die Namen dieser Ziele kann man der beim LaTeX-Durchlauf entstandenen `.out`-Datei entnehmen. Im Beamer-Paket sind für die Navigation eine ganze Reihe von Hyperlink-Befehlen definiert. Bei den meisten Themen werden diese automatisch eingebunden und bieten diverse Navigationsmöglichkeiten per Mausklick.

Anstelle des textorientierten Links bietet das Beamer-Paket auch ein paar vordefinierte Buttons:

`\beamerbutton{Buttontext`: Ovaler, farbig unterlegter Button
`\beamergotobutton{Buttontext`: Dito mit Rechtspfeil
`\beamerskipbutton{Buttontext`: Dito mit doppeltem Rechtspfeil
`\beamerreturnbutton{Buttontext`: Dito mit Linkspfeil
`\beamerbutton{Buttontext`: Ovaler, farbig unterlegter Button
`\beamerbutton{Buttontext`: Ovaler, farbig unterlegter Button

Die Buttons werden anstelle des beschreibenden Textes bei `\hyperlink` eingesetzt.

Auf die gleiche Art und Weise kann auf externe Dokumente oder Adressen im Web verwiesen werden. Zu diesem Zweck gibt es die Befehle `\href` und `\url`. Beispiel:

```
Auf der \href{http://www.netzmafia.de/}{Website} finden Sie ...
```

9.2.5 Überblendeffekte

Das Beamer-Paket unterstützt die Überblendeffekte, die der Acrobat Reader beherrscht – es werden also nur die passenden PDF-Konstrukte durchgereicht. Deshalb ist mit solchen Effekten Vorsicht geboten, denn eventuell kann nicht jeder PDF-Betrachter alle darstellen. Außerdem kostet das Überblenden Zeit und wird auch schnell etwas nervig. Bei fast allen Effekten können Dauer und Richtung des Effekts als Optionen der Form `duration=sekunden` bzw. `direction=winkel` angegeben werden, wobei für den Winkel nur die Werte 0, 90, 180, 270 und für den „glitter“-Effekt 315 zulässig sind. Unter anderem gibt es folgende Effekte:

PDF-Überblendeffekte

`transdissolve`
`transboxout` und `transboxin`
`transblindshorizontal` und `transblindshorizontal`
`transsplithorizontalin` und `transsplithorizontalout`
`transsplitverticalin` und `transsplitverticalout`
`transglitter`
`transwipe`

9.2.6 Handouts etc.

Leider reicht der Platz hier nicht aus, die zahllosen Möglichkeiten des Beamer-Pakets umfassend zu beschreiben (immerhin ist die Originaldokumentation über 100 Seiten lang). Deshalb abschließend noch ein Tipp, wie Sie eine Papierversion für die Zuhörer erstellen können. Durch die Zeile

```
\documentclass[handout, ...]{beamer}
```

werden jeweils die Endzustände der interaktiven Folien dargestellt und man kann einen Ausdruck erzeugen. Beachten Sie das Seitenformat von nur 128 mm x 196 mm; beim Drucken ist da ein Vergößerungsfaktor zu wählen.

Genauso einfach ist die Produktion von Overhead-Folien (falls mal kein Beamer zur Verfügung steht). Da wird dann anstelle von „handout“ ganz einfach „trans“ eingesetzt. Übrigens sollte man für alle Fälle bei wichtigen Vorträgen auch einen Satz Folien mitnehmen – falls die Technik doch mal versagt.

10

LaTeX-Makros schreiben

Um das Erstellen eigener LaTeX-Befehle (Makros) kommen Sie früher oder später nicht herum, denn zum Einen können Sie kleine Makros als Abkürzung für lange Befehle schreiben, zum Anderen können Sie eine umfangreiche Eingabe mehrerer Befehle auf einen einzigen Befehl reduzieren. Dabei können Sie dann sogar Parameter übergeben und es gibt sogar optionale Argumente. Fangen wir aber mal ganz einfach an.

10.1 Definition neuer Kommandos

Die einfachste Art, ein Makro zu definieren, ist die folgende:

```
\newcommand{\befehl}{Makroinhalt}
```

Der neu zu definierende Befehl darf dabei noch nicht existieren. Gibt es den Befehl schon, kann er mit dem Befehl `\renewcommand` umdefiniert werden, der im Übrigen genauso arbeitet wie `\newcommand`.

Die Angabe des Makro-Inhaltes kann sowohl aus Text als auch aus beliebigen LaTeX-Befehlen bestehen. Ein Beispiel:

```
\newcommand{\ddk}{\textbf{Donauampfschiffahrtsgesellschaftskapitän}}
...
Auf der Kommandobrücke traf er auf den \ddk.
```

Schön wäre es, wenn der Text, der von dem Makro ausgegeben wird, variabel gestaltet werden könnte. Genau dazu können Sie die Makro-Parameter verwenden. Dazu wird der Befehl `\newcommand` um einen optionalen Parameter erweitert, der angibt, wie viele Parameter das Makro selbst haben soll:

```
\newcommand{\befehl}[Parameteranzahl]{Makroinhalt}
```

Sie müssen beim Aufruf Ihres Makros diese Anzahl von Parametern auch unbedingt mit angeben. Innerhalb des Makro-Inhaltes können die Parameter mithilfe von #1 #2 usw. angesprochen werden. Insgesamt können bis zu neun Parameter benutzt werden. Dazu ein Beispiel:

```
\newcommand{\bfit}[1]{\textbf{\textit{#1}}}
```

sorgt dafür, dass der übergebene Parameter fett und kursiv gedruckt wird. Sie können beispielsweise das Makro `\bfit{folgendermaßen}` benutzen.

Manchmal sind auch recht profane Makros wie die folgenden ganz nützlich, mit denen im Buch Kommandos bzw Dateiangaben gesetzt werden:

```
\newcommand{\komm}[1]{{\tt #1}}  % Schriftart für Unix-Kommandos
\newcommand{\dat}[1]{{\tt #1}}   % Datei- und Verzeichnisnamen
```

Natürlich könnte man das auch mittels `{\tt ...}` erledigen, aber wenn man nun auf die Idee käme, statt der Schreibmaschinenschrift Kursivschrift oder Kapitälchen zu verwenden, müsste man sich durch das gesamte Manuskript arbeiten, denn globales Suchen/Ersetzen würde ja auch andere Textteile betreffen, die in Schreibmaschinenschrift bleiben sollen. Die Verwendung von Makros macht solche Änderungen sehr leicht. Eine weite Anwendung liegt darin, bei der Rohfassung zunächst nur ein sehr einfaches Makro zu definieren, was dann später aufgemotzt werden kann.

Wenn Sie mit eigenen Makros experimentieren, kann es vorkommen, dass sich bei deren Anwendung unerwünschte Leerzeichen zeigen. Das rührt meist von Leerzeichen oder Zeilenumbrüchen bei der Definition des Makros her. Schreiben Sie daher Ihre Makros so kompakt wie möglich. Die Zeilenwechsel lassen sich durch ein %-Zeichen am Ende der Zeile „entschärfen“.

Im folgenden einfachen Beispiel sollen verschieden große quadratische Kästchen definiert werden. Die Namensgebung lehnt sich an die der Schriftgrößen an:

```
\newcommand{\bull}{\rule{0.8ex}{0.8ex}}
\newcommand{\Bull}{\rule{1ex}{1ex}}
\newcommand{\BULL}{\rule{1em}{1em}}
```

Damit kann man nun die folgenden Kästchen ausgeben:
`\bull` ▪ `\Bull` ■ `\BULL` ■
und in `\Large`: \bull ▪ \Bull ■ \BULL ■ .

Probieren wir noch ein Beispiel: Es soll ein Formular gestaltet werden, das bei Eingabefeldern Kästchen für jeden Buchstaben bzw. jede Ziffer hat. Da können wir das verwenden, was auf Seite 35 bei den Boxen erwähnt wurde:

```
\newsavebox{\KKBox}% Neue Box benennen und definieren
% \rule legt die Höhe, \hspace die Breite des Kästchens fest
\sbox{\KKBox}{\fbox{\rule{0mm}{1em}\hspace{1ex}}}
\newcommand{\KK}{\usebox{\KKBox}}

% dann eine Eingabeanforderung:
Bankleitzahl: \KK\KK\KK\quad\KK\KK\KK\quad\KK\KK \\
Kontonummer: \KK\KK\KK\quad\KK\KK\KK\quad\KK\KK\KK\quad\KK\KK\KK \\
```

Das Ergebnis stellt sich dann folgendermaßen dar:

Bankleitzahl: ☐☐☐ ☐☐☐ ☐☐
Kontonummer: ☐☐☐ ☐☐☐ ☐☐☐ ☐☐☐

Möchten Sie ein eigenes Makro definieren, bei dessen Aufruf der erste Parameter optional ist, also in eckigen Klammern angegeben wird, gehen Sie folgendermaßen vor:

```
\newcommand{\befehl}[Parameterzahl][Default]{Makroinhalt}
```

Bei der Parameterzahl wird der optionale Parameter natürlich mitgezählt. Wird beim Aufruf des Makros der optionale Parameter angegeben, so wird er Platzhalter #1 zugewiesen, alle anderen Parameter den Platzhaltern #2, #3 usw. Im anderen Fall nimmt #1 den Default-Wert an. Auch dazu ein Beispiel:

```
\newcommand{\Name}[3][Herr]{#1 #2 {\bf #3}}

...
\Name{Donald}{Duck},
\Name{Dagobert}{Duck},
\Name[Frau]{Daisy}{Duck}
...
```

Das ergibt:

... Herr Donald **Duck**, Herr Dagobert **Duck**, Frau Daisy **Duck** ...

Problematisch kann auch die Verwendung des Mathematik-Modus sein, denn das Makro kann ja im normalen Absatz-Modus genauso gut aufgerufen werden, wie in der Mathematik-Umgebung. Abhilfe bietet hier das Makro `\ensuremath`. In der normalen Umgebung sorgt es dafür, das in den Mathe-Modus umgeschaltet wird (und nacher wieder zurück), ansonsten macht es gar nichts. Der neue Befehl `\PYT` im folgenden Beispiel kann sowohl im Mathe-Modus als auch im Absatz-Modus verwendet werden:

```
\newcommand{\PYT}{\ensuremath{a^2+b^2=c^2}}
...
\[ \PYT \]
...
Pythagoras hat herausgefunden: \PYT
```

liefert in beiden Fällen die richtige Formelausgabe:

...

$$a^2 + b^2 = c^2$$

...Pythagoras hat herausgefunden: $a^2 + b^2 = c^2$

10.2 Die Definition eigener Umgebungen

Als Umgebung oder Environment bezeichnet man in LaTeX Befehlsumgebungen, die mit `\begin{...}` und `\end{...}` geklammert werden. Mit dem Befehl

```
\newenvironment{name}[parameteranzahl]{begin-befehle}{end-befehle}
```

können Sie sich solche Umgebungen einschließlich Parameterübergabe selbst definieren. Für die Parameter gelten die gleichen Regeln wie bei der Neudefinition von Kommandos.

Legen Sie einen neuen Namen für Ihre Umgebung fest und definieren Sie anschließend die Befehle, die jeweils beim Eintritt (`\begin{...}`) und Austritt (`\end{...}`) der Umgebung ausgeführt werden sollen. Zu übergebende Parameter müssen beim Eintreten in die Umgebung angegeben werden. Ihre Umgebung können Sie dann mittels `\begin{Name}{Parameter}` ... `\end{Name}` benutzen. Ein Beispiel:

```
% Umgebung fuer Listings und andere Textteile, die 'verbatim'
% ausgegeben werden sollen. Schrift kleiner (footnotesize),
% TT-Font, Abstand zum Text etwas kleiner als bei Standard-verbatim.
\newenvironment{listing}
 {\vspace{-6pt}\footnotesize\verbatim}% begin-Befehle
 {\endverbatim\vspace{-6pt}}% end-Befehle
```

Das Beispiel macht sich auch zunutze, dass bei der `verbatim`-Umgebung die Makros `\verbatim` (für `\begin{verbatim}`) und `\endverbatim` (für `\end{verbatim}`) zur Verfügung stehen.

Ein weiteres Beispiel soll das Umfeld einer Tabelle vereinfachen. Der neuen Umgebung `tabelle` werden drei Parameter mitgegeben, die Tabellenspalten-Definition, die Tabellenüberschrift (caption) und ein Label, damit man im Text auf die Tabelle verweisen kann:

```
\newenvironment{tabelle}[3]%
 {\begin{table}[h!t]
  \renewcommand{\arraystretch}{1.20}% Abstände etwas groesser
  \begin{center}
  \caption{#2 \label{#3}}
  \vspace{3pt}
  \begin{tabular}{#1}}% ende begin-Befehle
 {\end{tabular}
  \end{center}
  \end{table}}%
```

10.3 Umdefinition von Befehlen und Umgebungen

Auch hier gelten die Regeln wie bei der Neudefinition von Makros. Mit Vorsicht ist jedoch an das Umdefinieren von LaTeX-Befehlen und -Umgebungen heranzugehen. Eine Umdefinition eigener Kommandos ist natürlich problemlos. Die Kommandos zur Umdefinition lauten `\renewcommand` und `\renewenvironment`.

Sie haben den gleichen syntaktischen Aufbau und unterliegen den gleichen Regeln wie die Kommandos zur Neudefinition von Makros und Umgebungen. Jedoch muss als Name ein bereits existierendes Kommando eingegeben werden. Sinnvoll und erwünscht sind zum Beispiel die Umdefinitionen von `\baselinestretch`, `\arraystretch` und anderen Größenangaben, zum Beispiel:

```
% Darstellung der Bilder im Text aendern
% Bildunterschriften einen Punkt kleiner setzen
\renewcommand{\captionfont}{\small}
\renewcommand{\captionlabelfont}{\small \bf}
% Bildunterschriften mit 'Bild' statt 'Abbildung'
\renewcommand{\figurename}{Bild }
```

10.4 Der \newtheorem-Befehl

`\newtheorem` erzeugt eine Umgebung für Lehrsätze (Theoreme), Definitonen usw. Bei diesem Befehl gibt es gleich drei Syntaxvarianten:

```
\newtheorem{Name}{Überschrift}
\newtheorem{Name}{Überschrift}[KapZähler]
\newtheorem{Name}[ThmZähler]{Überschrift}
```

Der Aufruf im Text erfolgt wie bei anderen Umgebungen: `\begin{Name} ... \end{Name}`. Es wird eine abgesetzte Umgebung mit Überschrift und einem Zähler erzeugt – wie beispielsweise bei Kapiteln und Abschnitten. Das Verhalten des Zählers hängt von den optionalen Parametern *KapZähler* bzw. *ThmZähler* ab, deren Wirkung sich unterscheidet:

- Ohne Verwendung der optionalen Zähler, werden alle Vorkommen der `theorem`-Umgebungen der Reihe nach durchnummeriert (Satz 1, Satz 2 usw.).
- Bei Angabe des Parameters *KapZähler* (z. B. chapter, section usw.) wird die Nummer der Umgebung zurückgesetzt, wenn der entsprechende Gliederungsbefehl aufgerufen wird. Man kann die Theoreme also kapitel- oder abschnittsweise durchzählen.
- Bei Angabe des Parameters *ThmZähler* (der Name einer vorher definierten anderen `theorem`-Umgebung) erhält die neue Umgebung keine eigene Nummerierung, sondern wird mit der angegebenen Umgebung mitgezählt.

Beispiel:

```
\newtheorem{these}{These}
\newtheorem{folg}{Folgerung}[chapter]

\begin{these}[Plate]
Wir können machen, was wir wollen, die Unordnung nimmt zu.
(Abgeleitet aus dem 2. Gesetz der Thermodynamik)
\end{These}

\begin{folg}
Aufräumen führt nur zu Unordnung an einer anderen Stelle
im Universum.
\end{folg}
```

Das ergibt:

These 1 (Plate) *Wir können machen, was wir wollen, die Unordnung nimmt zu. (Abgeleitet aus dem 2. Gesetz der Thermodynamik)*

Folgerung 1 *Aufräumen führt nur zu Unordnung an einer anderen Stelle im Universum.*

LaTeX bietet eine Fülle weiterer Möglichkeiten, unter anderem das Erstellen von Zeichnungen oder exzessive Möglichkeiten der Textgestaltung. Wir wollen jedoch an dieser Stelle mit der Einführung in die Arbeit mit LaTeX schließen, denn erstens kann das Buch nicht beliebig dick werden und zweitens bietet das bisher Gesagte mehr als genug Grundlagen für alle studentischen Belange.

Erlauben Sie uns noch ein paar Schlussbemerkungen, bevor wir auf die Tools zur Bearbeitung des Quelltextes eingehen:

- Texte ohne WYSIWYG zu erstellen, erfordert ein gewisses Umdenken. Sie werden aber nach einiger Zeit merken, dass die Gedanken viel schneller und klarer „zu Papier“ gelangen, wenn man sich erst mal nicht um das Aussehen kümmern muss.

- Halten Sie durch! Zu Beginn ist die Frustration, die durch die scheinbar unverständlichen Fehlermeldungen erzeugt wird, relativ hoch. Mit der Zeit werden die Fehler weniger und Sie verstehen auch, was LaTeX Ihnen sagen will.
- Aus den Quelltexten können Sie auch in zehn Jahren noch ein Buch oder einen Artikel weiterbearbeiten und in hoher Qualität ausdrucken – versuchen Sie das mal mit den Dateien, die mit einer beliebigen Textverarbeitung erstellt wurden.
- Schlussendlich bietet Ihen LaTeX die Möglichkeit, den Quelltext per Programm zu erzeugen. Sie können also per Programm absolut „schicke“ Dokumente generieren.

11

Metafont- und PostScript-Schriften

Der Umgang mit Schriftarten bereitet bei der Arbeit mit LaTeX vermutlich die größten Verständnisprobleme. Dieser Abschnitt versucht, ein wenig Klarheit zu schaffen. Die zwei zentralen Themen dieses Abschnitts sind das LaTeX-Zusatzprogramm `metafont`, mit dem die LaTeX-eigenen Schriften erzeugt werden, und die Verwendung von PostScript-Schriften an Stelle der LaTeX-Originalschriften. Noch viel mehr Informationen über die Hintergründe von MetaFont- und PostScript-Schriften finden Sie in den deutschen TeX-FAQs unter *http://www.dante.de/faq/de-tex-faq/*.

Wahrscheinlich kommt in einem Dokument nicht nur ein Schriftschnitt vor, sondern Sie wollen einzelne Wörter irgendwie hervorheben – z. B. mit **fetter**, *kursiver* oder grosser Schrift. Eventuell werden wichtige Namen in KAPITÄLCHEN gedruckt. Das ist mit LaTeX natürlich kein Problem. [1]

Folgende Schrift-Auszeichnungen sind möglich:

`\textrm{...}`	roman (serif)	`\textsf{...}`	sans serif
`\texttt{...}`	`typewriter`	`\textsc{...}`	SMALL CAPS
`\textit{...}`	*italic*	`\textbf{...}`	**bold face**
für normale Hervorhebungen wird i.d.R. `emph{...}` (*emphasized*) verwendet			

Für Code-Beispiele eignet sich die `verbatim`-Umgebung. Sie gibt in Schreibmaschinenschrift alles genau so wieder, wie es im LaTeX-File steht, ohne eventuell im Text vorhandene Befehle auszuführen. Was zwischen `\begin{verbatim}` und `\end{verbatim}` steht, wird 1:1 gedruckt. Die Kurzform `\verb?...?` wird für kurze Stellen im Fließtext verwendet. Das „?“ dient als Delimiter. Sollte ein „?“ im auszugebenden Text vorkommt, nehmen Sie ein anderes Zeichen als Delimiter – außer Buchstaben, * und Leerzeichen ist alles erlaubt.

Auch die Schriftgröße lässt sich verändern: Sie könenn entweder in der Präambel den Befehl `\documentclass[...]{...}` mit der Option `[10pt]` (Standard), `[11pt]` oder `[12pt]` aufrufen oder Sie wählen mit einem der folgenden Befehle die Schriftgröße:

[1] Bevor Sie jedoch nun Schriftschnitte und -grössen munter mischen, sollten Sie bedenken, dass LaTeX in den Standardeinstellungen bereits für optimale Lesefreundlichkeit und ein ansprechendes Schriftbild sorgt. Auch werden selten mehr als zwei oder drei Schriftarten benötigt.

`\tiny`	winzig
`\scriptsize`	sehr klein
`\footnotesize`	ziemlich klein
`\small`	klein
`\normalsize`	normal gross
`\large`	ein bisschen grösser
`\Large`	ziemlich gross
`\LARGE`	gross
`\huge`	sehr gross
`\Huge`	gigantisch

Der Befehl zur Änderung der Schriftgröße kann auf drei verschiedene Arten verwendet werden:

- mit `{\Large ...}`, wenn nur wenige Worte vergrößert werden sollen (innerhalb der geschweiften Klammern)
- innerhalb von `\begin{Large}` und `\end{Large}` für einen längeren Text (z. B. einen Absatz)
- `\Large ...` für den gesamten Text ab diesem Aufruf bis zum nächsten Textgrössen-Änderungsbefehl

Sollte Ihnen die LaTeX-Standardschrift (Computer Modern) nicht gefallen, können Sie das Dokument auch in anderen Schriftarten setzen. Auf die etwas komplizierte Verwendung von *beliebigen* Postscript- oder TrueType-Schriften gehe ich hier nicht ein, oftmals ist das auch nicht nötig.

Der Befehl `\usepackage{schrift}` in der Präambel, wobei `schrift` eine der untenstehenden Namen annehmen kann, ersetzt die LaTeX-Standardschriftarten im gesamten Dokument:

Schrift	**Serifenschrift**	**Groteskschrift** (`\textsf{...}`)
`avant`	Computer Modern	Avant Garde
`avantgar`	Avant Garde	Computer Modern Sans
`bookman`	Bookman	Avant Garde
`chancery`	Zapf Chancery	Computer Modern Sans
`charter`	Charter	Computer Modern Sans
`courier`	Computer Modern	Computer Modern Sans (texttt wird ersetzt)
`helvet`	Computer Modern	Helvetica
`newcent`	New Century Schoolbook	Avant Garde
`palatino`	Palatino	Helvetica
`times`	Times	Helvetica
`zapfchan`	Zapf Chancery	Computer Modern Sans

Mit `\usepackage{mathptmx}` werden nicht nur das Dokument, sondern auch darin enthaltene Formeln in Times gesetzt, `\usepackage{mathpazo}` bewirkt das gleiche mit der Palatino. Bei den anderen Schriften wird immer Computer Modern für Formeln verwendet, was in der Kombination nicht immer schön aussieht. Für weitere Experimente mit Schriftarten sei auf die Dokumentationen der Befehle `\usefont` bzw. `\fontfamily{...}\selectfont` verwiesen.

11.1 Metafont-Schriften

Als LaTeX ursprünglich entwickelt wurde, war der Druckermarkt noch unübersichtlicher als jetzt. Einen etablierten Standard wie PostScript mit zahllosen vordefinierten Schriftarten zu einem für (beinahe) jedermann erschwinglichen Preis gab es damals noch nicht. Aus diesem Grund wurde gleichzeitig mit LaTeX das Programm `metafont` entwickelt. Dieses Programm ist für die Berechnung der Schriften zuständig. Das Ziel war es, möglichst jeden beliebigen Drucker in optimaler Qualität unterstützen zu können. Vor diesem Hintergrund ist auch zu verstehen, warum `metafont` mit Bitmap-Schriften (und nicht wie PostScript mit beliebig skalierbaren Vektorschriften) arbeitet. LaTeX und das

Metafont-System bilden bis heute ein integratives Paket; jedes Programm für sich ist praktisch wertlos.

Wie funktioniert Metafont? Ausgangspunkt für alle Schriften sind *name.mf*-Dateien. Diese Textdateien enthalten Kommandos zum Zeichnen der einzelnen Buchstaben einer Schriftart. Das Programm `metafont` erzeugt je nach den angegebenen Optionen eine oder zwei Dateien: in jedem Fall eine Bitmap-Datei *name.nnnpk* und manchmal (wenn diese Datei noch nicht existiert) die Metrikdatei *name.tfm*.

Die Bitmap-Dateien (`*.nnnpk`) enthalten die Schrift in komprimierter Form. In dieser Bitmap ist jeder Buchstabe durch Tausende von Einzelpunkten (Pixeln) dargestellt. *nnn* steht dabei für einen Faktor aus Auflösung in dpi (dots per inch) und Vergrößerung. Typische dpi-Werte sind 300 oder 600 dpi bei Laserdruckern sowie 1270 oder 2540 dpi bei Belichtungsgeräten. Der Vergrößerungsfaktor kommt ins Spiel, wenn eine Schrift in einer anderen Größe als in ihrer Entwurfsgröße benötigt wird (z. B. bei `\small` oder `\large`). Daraus ergeben sich dann Werte wie 720 (600 mal 1,2).

Die Metrikdateien (*`*.tfm`*) enthalten alle Angaben über die Größen der einzelnen Buchstaben. Auch wenn es zu einer Schrift mehrere *`*pk`*-Dateien gibt, existiert immer nur eine Metrikdatei (die wahre Größe der Buchstaben ist ja von der Druckerauflösung unabhängig).

Nun zur realen Bedeutung dieser beiden Dateitypen: Die Metrikdateien *`*.tfm`* werden während der Bearbeitung eines Textes durch LaTeX benötigt. LaTeX entnimmt diesen Dateien die Information, wie groß die jeweiligen Buchstaben sind, und führt anhand dieser Daten den Zeilen- und Seitenumbruch durch. Das Ergebnis ist eine DVI-Datei (*device independent*), die die eigentlichen Schriften nicht enthält und in dieser Form weder ausgedruckt noch angezeigt werden kann.

Die *`*pk`*-Dateien werden erst beim Ausdruck bzw. bei der Anzeige der Datei am Bildschirm (`xdvi`) benötigt. Wenn die gerade erforderliche *`*pk`*-Datei in der Auflösung des Druckers bzw. Bildschirms und in der gewünschten Vergrößerung noch nicht existiert, wird `metafont` automatisch gestartet. Deswegen kann es beim ersten Ausdruck eines Textes mit vielen Schriftarten und -größen zu erheblichen Verzögerungen kommen. Beim nächsten Mal stehen die erforderlichen *`*pk`*-Dateien dann aber bereits zur Verfügung.

Im Regelfall müssen Sie sich nicht um Schriftartdateien kümmern und brauchen das Programm `metafont` (Kommandoname `mf`) nie selbst aufzurufen. Alle erforderlichen *`*.mf`*- und *`*.tfm`*-Dateien sind bereits vorinstalliert und die *`*.nnnpk`*-Dateien werden je nach Bedarf automatisch erzeugt.

CM-Schriften: Unter LaTeX kommen per Default vier Familien der CM-Schriften zum Einsatz: CM Roman (Standardschrift), CM Sans Serif, CM Typewriter und eine CM-Roman-ähnliche Schrift für mathematische Zeichen. (CM steht für *computer modern*. Die CM-Schriften wurden vom LaTeX-Erfinder und -Entwickler Donald Knuth entworfen.)

Die CM-Schriften bestehen nur aus den 128 Zeichen des ASCII-Zeichensatzes (bzw. ISO-7-Bit). Buchstaben wie å, é, ñ oder ö sind keine eigenen Zeichen, sondern werden durch die Überlagerung unterschiedlicher Zeichen gebildet. Die Zuordnung zwischen Zeichen und Codes wird als OT1-Kodierung bezeichnet.

Da dies typografisch nicht optimal war, wurden die **EC-Schriften** geschaffen, in denen fast alle in europäischen Ländern üblichen Buchstaben als eigene Zeichen enthalten sind (insgesamt 256 Zeichen). Um diese Schriften zu nutzen, müssen Sie die Zeile `\usepackage[T1]{fontenc}` in Ihr LaTeX-Dokument einfügen.

Diese Anweisung verändert rein optisch fast nichts am Aussehen Ihrer mit LaTeX gesetzten Dokumente. Sie müssten schon ein Typografie-Experte sein, um Unterschiede zu bemerken. Allerdings sind nun viele Zeichen in den DVI-Dateien anders codiert (T1-Kodierung). Das gilt auch für daraus resultierende PDF-Dateien. Ein wesentlicher Vorteil der T1-Kodierung besteht darin, dass Sie nun in PDF-Dokumenten auch nach Wörtern mit den Buchstaben ä ö ü ß oder anderen Buchstaben außerhalb des ASCII-Zeichensatzes suchen können.

Manche häufig in Texten vorkommende Zeichen fanden allerdings auch in den EC-Schriften nicht Platz. Als Ergänzung zu den EC-Schriften wurden daher die **TC-Schriften** entworfen, die zusätzliche Zeichen enthalten. Für die Zuordnung zwischen Zeichen und Kodes gilt die TS1-Kodierung. Um auch diese Schriften nutzen zu können, fügen Sie die Zeile `\usepackage{textcomp}` in Ihr LaTeX-Dokument ein. Es stehen Ihnen nun eine Reihe neuer `\textxxx`-Kommandos für diverse Sonderzeichen zur Verfügung.

11.2 PostScript-Schriften (Type-1-Fonts)

Die auf Metafont basierenden Bitmap-Schriften waren zwar 1985 (LaTeX 2.09) richtungsweisend, sie sind heute aber nicht mehr zeitgemäß. Zur optimalen Weiterverarbeitung von LaTeX-Texten ist es wünschenswert, die Bitmap-Schriften durch PostScript-Schriften (so genannte Type-1-Fonts) zu ersetzen. Das macht qualitativ optimale Ausdrucke möglich (etwa für den Buchdruck) und erleichtert die Weitergabe im PDF-Format ohne Pixelartefakte.

PostScript-Ersatz für die CM-Schriften: Die aktuelle teTeX-Version enthält für die CM-Schriften äquivalente PostScript-Schriften (Verzeichnis `/usr/share/texmf/fonts/type1/bluesky/`). Diese Schriften werden beim Aufruf von `dvips` automatisch eingesetzt. Insofern ist das Problem der Bitmap-Schriften eigentlich schon gelöst, ohne dass Sie irgendetwas tun müssen. Die PostScript-Schriften sind durch eine Umwandlung der originalen Metafont-Schriften entstanden und sollten absolut identisch aussehen.

Das einzige Problem an dieser scheinbar optimalen Lösung besteht darin, dass es momentan noch keinen PostScript-Ersatz für die EC- und TC-Schriften gibt. Deswegen funktioniert die automatische Ersetzung der LaTeX-Schriften durch PostScript-Schriften *nicht*, wenn sich die beiden folgenden Zeilen im LaTeX-Dokument befinden:

```
\usepackage[T1]{fontenc} % T1-Codierung statt OT1-Codierung
\usepackage{textcomp}    % Sonderzeichen mit TS1-Codierung
```

PostScript-Varianten für die EC- und TC-Schriften sind im Internet bereits verfügbar. Sie werden wahrscheinlich in die teTeX-Distribution aufgenommen, sobald sie vollständig ausgereift sind.

PostScript-Standardschriften einsetzen: Ein anderer Lösungsweg besteht darin, auf die CM-Schriften ganz zu verzichten und stattdessen auf PostScript-Standardschriften wie Times, Palatino, Helvetica oder Courier umzusteigen. Erfreulicherweise gibt es zu diesem Zweck fertige LaTeX-Pakete (Sammelbegriff PSNFSS2e). Sie müssen lediglich ein paar `\usepackage`-Anweisungen in Ihr LaTeX-Dokument einfügen. Beachten Sie aber, dass sich dadurch der Zeilen- und Seitenumbruch Ihres Dokuments ändern kann.

```
\usepackage{mathpazo}    % Palatino statt der LaTeX-Originalschrift
```

In LaTeX-Dokumenten gibt es immer vier Schriftfamilien (Standardschrift, Sans Serif, Typewriter, Mathematik-Schrift). Die folgenden Pakete ersetzen aber jeweils nur einzelne Schriften und lassen die anderen unverändert. Wenn Sie alle LaTeX-Schriften durch PostScript-Schriften ersetzen möchten, müssen Sie mehrere Pakete kombinieren (z. B. *mathptmx*, *helvet* und *courier*)!

PostScript-Font-Pakete	
avant	AvantGarde statt Sans Serif, andere Schriften: Original-LaTeX
bookman	Bookman als Standardschrift, AvantGarde statt Sans Serif, Courier statt Typewriter
chancery	Zapf Chancery als Standardschrift, andere Schriften: Original-LaTeX
charter	Charter als Standardschrift, andere Schriften: Original-LaTeX
courier	Courier statt Typewriter, andere Schriften: Original-LaTeX
helvet	Helvetica statt Sans Serif, andere Schriften: Original-LaTeX
mathptmx	Times als Standardschrift, Times-ähnliche Mathe-Schrift, andere Schriften: Original-LaTeX
mathpazo	Palatino als Standardschrift, Palatino-ähnliche Mathe-Schrift, andere Schriften: Original-LaTeX
newcent	Standardschrift New Century Schoolbook, AvantGarde statt Sans Serif, Courier statt Typewriter
utopia	Utopia als Standardschrift, andere Schriften: Original-LaTeX

Das Paket *mathptmx* unterstützt keine Formeln in fetter Schrift (d. h. `\boldmath` steht nicht zur Verfügung).

Die Schriften Utopia und Charter sind zwar frei verfügbar, zählen aber nicht zu den 35 PostScript-Standardschriften von Adobe. Es kann daher sein, dass diese Schriften extra installiert werden müssen.

Um das Ergebnis weiter zu verbessern, sollten Sie bei den meisten Schriften den Durchschuss (den Zeilenabstand) mit `\linespread{n}` optimieren. Dabei ist *n* ein Faktor, der den Zeilenabstand vergrößert oder verkleinert. Bei vielen Schriften verbessert ein Wert zwischen 1.05 und 1.1 die Lesbarkeit.

Weil die Schrift Helvetica etwas größer als die meisten anderen Schriften ist, wird oft empfohlen, diese Schrift relativ zu den anderen Schriften ein wenig zu verkleinern. Dazu verwenden Sie die folgende Anweisung:

```
\usepackage[scaled=0.95]{helvet}
```

Die PostScript-Standardschriften stehen sowohl in der LaTeX-Original-Kodierung OT1 als auch in der erweiterten Kodierung T1/TS1 (wie bei den EC- und TC-Schriften) zur Verfügung. Die Dokumentation empfiehlt ausdrücklich, mit den beiden folgenden Kommandos die erweiterte Kodierung zu verwenden:

```
\usepackage[T1]{fontenc} % T1-Codierung statt OT1-Codierung
\usepackage{textcomp}    % Sonderzeichen mit TS1-Codierung
```

Wir haben allerdings die Erfahrung gemacht, dass PDF-Dokumente bei der T1-Kodierung im Adobe Reader manchmal schlechter lesbar sind als bei der Original-Kodierung OT1. Experimentieren Sie gegebenenfalls selbst!

Eine Eigenheit bei der Verwendung von PostScript-Schriften besteht darin, dass die beiden Apostrophe ' und ‘ nur mit einer Lupe voneinander zu unterscheiden sind. Typografisch mag das durchaus korrekt sein, wenn die Zeichen aber in Programmlistings benötigt werden, ist das ein großes Problem. Eine Notlösung besteht darin, den nach rechts gerichteten Apostroph durch das Mathematikkommando `$\grav$` zu bilden (liefert `).

Beliebige PostScript-Schriften nutzen: Während die Verwendung der PostScript-Standardschriften unkompliziert ist, erfordert die Nutzung anderer PostScript-Schriften einigen Aufwand: Die Schriften müssen so installiert werden, dass LaTeX, `dvips` und GhostScript sie finden. Hier fehlt allerdings der Platz, alle Details zu beschreiben. Stattdessen muss ich Sie auf die folgenden Internetseiten verweisen:

Kurzbeschreibung im Rahmen des Font-HOWTO-Dokuments:
http://www.tldp.org/HOWTO/Font-HOWTO/

Ausführliche, deutschsprachige Beschreibung von Christian Kuhn:
http://www.qno.de/computer/latex/fonts/tutorial.html

Ebenso ausführliche, englische Beschreibung von Matthew Amster-Burton:
http://www.mamster.net/tex/latex-fontfaq-amster-burton.pdf

12

Weiterführende Links

12.1 Online-Tutorials

ftp://tug.ctan.org/pub/tex-archive/info/beginlatex/beginlatex-3.6.pdf:
A beginner's introduction to typesetting with LaTeX von Peter Flynn

http://www.maths.tcd.ie/~Edwilkins/LaTeXPrimer/:
Getting Started with LaTeX von David A. Wilkins

http://www.ctan.org/tex-archive/info/lshort/german/l2kurz.pdf:
LaTeX2e-Kurzbeschreibung von Walter Schmidt, Jörg Knappen, Hubert Partl, Irene Hyna

http://www.ctan.org/tex-archive/info/lshort/english/lshort.pdf:
The Not So Short Introduction to LaTeX2e von Hubert Partl, Irene Hyna, Elisabeth Schlegl

http://www-math.cudenver.edu/~Ehgreenbe/booksEtc/SimplifiedIntro.html:
A Simplified Introduction to LaTeX von Harvey J. Greenberg

http://tug.ctan.org/tex-archive/info/latex4wp/latex4wp.pdf:
LaTeX for Word Processor Users von Guido Gonzato

ftp://ftp.fernuni-hagen.de/pub/pdf/urz-broschueren/broschueren/a0260003.pdf:
LaTeX - eine Einführung und ein bisschen mehr... von Manuela Jürgens

ftp://ftp.fernuni-hagen.de/pub/pdf/urz-broschueren/broschueren/a0279510.pdf:
LaTeX - Fortgeschrittene Anwendungen von Manuela Jürgens

12.2 Weitere Dokumente und Links

http://www.ctan.org/tex-archive/info/epslatex.pdf:
Using Imported Graphics in LaTeX and pdfLaTeX von Keith Reckdahl

http://www.tug.org/docs/pdf/gentle.pdf:
A Gentle Introduction to TeX von Michael Doob

http://tug.ctan.org/tex-archive/info/german/l2tabu/l2tabu.pdf:
Das LaTeX2e-Sündenregister von Mark Trettin

http://www.ibnm.uni-hannover.de/Mitarbeiter/beuerman/LaTeX2PDF.pdf:
Erstellen von leistungsfähigen PDF-Dokumenten mit LaTeX von Sascha Beuermann

http://www.p-joeckel.de/pdflatex/:
How-to make a PDF-document from a LaTeX-source von Patrick Jöckel

http://www.ctan.org/tex-archive/info/symbols/comprehensive/symbols-a4.pdf:
The Comprehensive LaTeX Symbol List von Scott Pakin

http://tex.loria.fr/general/downes-short-math-guide.pdf:
Short Math Guide for LaTeX von Michael Downes

http://texcatalogue.sarovar.org/:
Der TeX-Katalog

http://www.math.tu-dresden.de/~rudl/latex/fonts.pdf:
Schriftarten in LaTeX von Jan Rudl

Es folgt nun das Beispiel für ein Ganzseitenbild.

Stichwortverzeichnis